Traffic Analysis of the HTTP Protocol over TLS

George Danezis

University of Cambridge Computer Laboratory
William Gates Building, JJ Thomson Avenue
Cambridge CB3 0FD, United Kingdom
George.Danezis@cl.cam.ac.uk

Abstract. We analyze attacks that take advantage of the data length information leaked by HTTP transactions over the TLS protocol, in order to link clients with particular resources they might access on a web site. The threat model considered is a public news site that tries to protect the patterns of requests and submissions of its users by encrypting the HTTP connections using TLS, against an attacker that can observe all traffic. We show how much information an attacker can infer about single requests and submissions knowing only their length. A Hidden Markov Model is then presented that analyzes sequences of requests and finds the most plausible resources accessed. We note that Anonymizing systems such as the Safe Web service could be the victim of such attacks, and discuss some techniques that can be used to counter them.

1 Introduction

Privacy as a system property is sometimes confused with secrecy. Mechanisms such as SSL (Secure Socket Layer) [1] or its successor TLS (Transport Layer Security protocol) [2] have been created to protect the secrecy of HTTP information [3] being transmitted, but are commonly used to protect more general privacy properties. Services that offer anonymous browsing, such as Safeweb [5] and Anonymizer [6], use TLS encryption to hide the pages that their users access. This is not always an adequate protection of the client's privacy, as the traffic information not hidden by TLS still leaks a great deal of information.

To illustrate this, one can use as an example a public information service. This service works like a bulletin board: everyone can read all the submissions and everyone can submit comments or articles. At first glance there is no privacy problem associated with secrecy because no information is secret. This is not the case: users may wish to receive or submit information anonymously. Indeed for many systems such, as alcoholics anonymous or drug addiction discussion sites, this requirement is essential. In some cases users may want this anonymity to be close to absolute, so that even the web site administrators cannot trace them. To achieve this they must use an anonymous communication infrastructure such as MixMaster [7, 8] or Web Mixes [9]. In other cases assurance that no third party could reconstruct patterns of requests or submissions is sufficient. Many web site designers could be tempted to use end-to-end encryption, such as TLS,

to provide this assurance. Unfortunately, as we will show, it does not offer the expected protection.

In section 2 we shall introduce traffic analysis and the particularities of the TLS protocol. In section 3 we discuss the assumed threat model and potential sources of attacks. Section 4 and 5 present the measurements and attacks on single HTTP transactions and multiple transactions respectively. Finally section 6 proposes some countermeasures to protect web sites against the attacks.

2 Traffic Analysis and the TLS Protocol

Traffic analysis is the science of extracting information from meta data, or otherwise known as traffic data, produced by a communication. These include the routing data, length and timing of the communication stream. Recent work in this area includes using timing information to reduce the entropy of passwords sent using SSH [10] and guessing if a particular web page is already locally cached by a user [11]. Research into anonymous communication has also provided some insights about how the shape of traffic contained in a channel can be used to trace the communication. The onion routing project [12] presented strong evidence for the need to use dummy cover traffic in order to hide these patterns. Traffic analysis of HTTP transactions through anonymizing proxies has been mentioned in [13] and [14]. In [13] Hintz analyzes traffic packet lengths at the TCP level, in order to attack the SafeWeb [5] service. Previous work has also been presented about modeling user browsing on the web using Markov Models in [15].

Traffic analysis can be used to extract a variety of information. It can be used for *identification*, when the information extracted is used to find out who the sender of some data is, or which particular network card is active. It can also be used for *profiling* when the aim of the analysis is to extract some information about the target, such as their type or status. Finally traffic analysis can be used for *information extraction* when the objective of the analysis is to extract some of the information contained in a particular conversation.

The TLS protocol was designed to protect the secrecy and the authenticity of a data stream. Its specification [2] has an explicit warning about traffic analysis:

> "Any protocol designed for use over TLS must be carefully designed to deal with all possible attacks against it. Note that because the type and length of a record are not protected by encryption, care should be taken to minimize the value of traffic analysis of these values."

Indeed the basic building block of TLS, the TLS Ciphertext record, transmits the *Content type*, *Version* and *Length* of the record in clear, although the length could be easily reconstructed by an observer. In the case a block cipher is used to perform the encryption, the data is padded so that its length is a multiple of the block size. For each HTTPS [16, 17] request, one or more of these records are transmitted to the server and one or more of these records are received as a reply.

3 Threat Model & Attacks

We assume that all traffic between the web server, Sam, and the client, Claire, is transmitted using HTTPS (HTTP over TLS) [16, 17]. This connection can be either persistent or not: in a persistent connection all the requests and replies are sent over the same TCP (and TLS session), while non-persistent connections open a new TCP connection for each request and reply. The service performs some mixing on the items submitted so that the attacker can not just check which pages have changed in order to link users to items.

The attacker, Mallory, aims to link Claire's requests and replies with particular items that have been submitted to or sent out by Sam. Of course his ultimate aim may be beyond linking her to particular resources. Even decreasing the number of candidate pages for each request can be useful in the long run. He also may want to find out her interests, patterns of behavior, plans, people she knows etc. Because all the information is public Mallory knows the exact length of all the resources on the server, as he can use the server as any other user could. We also assume that the attacker can eavesdrop on all the communications between the clients and servers but cannot break the TLS encryption.

Mallory can gain information from the TLS protected HTTP communication in a variety of ways:

Information Gained by the Encrypted Client Requests The URL of different requested resources may have different lengths that are leaked. Most web clients will automatically send requests for components of web pages such as images or applets and these additional requests can be used to further narrow the choice of candidate pages requested by Claire. A technique that analyzes streams of resource lengths using Hidden Markov Models is presented in this paper. Other field lengths in the request header can also be used to perform traffic analysis. The referrer field indicates the previous URL accessed and is automatically inserted into requests. It is of variable length and contains information, that the attacker could use.

When Claire submits some information to Sam, she uses the POST method with the data she wants to submit as the contents of the message. Mallory can observe the length of the data submitted and use it later to link the submission with items on the site.

Information Gained by the Encrypted Server Replies In response to a request from Claire, Sam sends back a status line and, if the GET request was successful, the content of the resource requested follows. The content of the resources on Sam's server vary in length an Mallory can use this information to determine which information Claire is accessing. Since the information is public Mallory has a prior knowledge of the length of each resource and can try to link them.

Information Structure Attacks Mallory can observe that the information on Sam's site is organized in a particular manner. It could, for example, be using many pages linked together by hyper links. Most of the users would follow that structure and Mallory can use that assumption in order to refine

his guess about which pages Claire is accessing. A particular technique implementing this attack, based on Hidden Markov Models, is presented later in this paper.

Content Attacks The content of Claire's contribution is not visible to Mallory in transit, but all the messages are visible from the site. The message may contain some information that only Claire knows, either by chance or because it has been specifically constructed to be unique to Claire. Mallory can then scan all the messages on Sam's web site and look for this information, therefore linking the article to Claire. This is not a flaw of TLS and could be exploited even if the links between Sam and Claire were completely anonymous.

All the attacks described rely on data that Mallory has passively collected from the participants, or data publicly available though Sam's service. If the attacker is allowed to contribute to Sam's server he can devise active traffic analysis attacks, using appropriate baits.

The Chosen Submission Length Attack (Sting Attack) Mallory can contribute resources to Sam's service with a length engineered to differentiate them from other documents, and use this information to find out who would be interested in them. This technique can be used to fingerprint sites and pages. Many news services allow users to contribute comments that are appended to the articles, that can also be used to tag the sites.

The Chosen Submission Name Attack If Sam allows the contributors to specify part of the URL of the resources, Mallory can make it arbitrarily long (subject to the URL specification restrictions [4]) in order to enhance his ability to differentiate between GET requests.

External Resource Attack If Mallory can link external resources to the information he submits he can observe how these are accessed in order to discover what resource Claire is browsing. The external resource could be accessed automatically, as it is done for images, or manually by following links. This attack can be used to enhance the performance of the "information structure attacks" as described above.

4 Implementing Some Attacks

The initial observation that the length of the communication provides enough information to link it with particular resources is often received with skepticism. In order to test the hypothesis that information about the resource assessed can be extracted from the sizes of the requests, some experiments were performed. The first used downloaded web sites and the second used data extracted from log files to perform the analysis. As it will be presented in detail below, knowledge of the length of communications restricts seriously the number of candidate resources, and in many cases provides the attacker with the exact resource that was accessed.

4.1 Measuring the Information Contained in the Resource Lengths

The aim of the first experiment was to calculate the entropy, and therefore the information that can be extracted, from the length distribution of the resources on a web server. To do this the news site `uk.indymedia.org` [18] was used. The choice was not random. On the 8th of May 2001 an affiliated site was the recipient of an FBI notice demanding its log files. The second reason for this choice is that the site is very close to the model we assume. Clients can freely submit articles, comments are appended to articles, all articles and comments are public *etc.* Since the site is not very large, only a few hundred pages could be downloaded. The size of the resources was a very good indicator of each of them. Most of the resources could be uniquely identified by their size quantized to an 8 byte boundary, and the remaining were only confused with very few other resources.

The experiment was repeated for a larger news site `news.bbc.co.uk` [19]. A few thousand pages were downloaded and their sizes were quantized to an eight byte boundary. The size of the files provides around 9.8 bits of entropy which is enough to discriminate most of the pages. That means that using this information we could discriminate up to $2^{9.8}$ pages but in practice we should expect to see pages with colliding lengths after the service grows larger than $2^{4.9}$ resources. It was observed that the larger the files get, the easier it is to discriminate between them. This makes multimedia files more difficult to camouflage than normal HTML documents. The number of images automatically fetched was also used, to provide additional information about the files fetched. The number of images fetched could be available to an attacker if non-persistent HTTP connections are used by the server, as each image opens a new TCP socket. The entropy given by the number of images was around 5 bits, but if the attacker already knows the length of the file this information only provides 0.3 additional bits of information.

4.2 Assessing the recognition rate using resource length

The second experiment was performed on the logs of the Cambridge Student Run Computing Facility `www.srcf.ucam.org` [20]. An analysis was done using HTTP logs to see how many requests could be attributed to particular resources, only taking into account their lengths. It gave the following results.

- 35.6% of the resource lengths can be attributed to fewer than 10 URI's
- 66.3% of the resource lengths can be attributed to fewer than 20 URI's
- 81.7% of the resource lengths can be attributed to fewer than 40 URI's

Figure 1 shows that the number of requests comes down linearly as the uncertainty about which page was accessed (in bits) increases. The horizontal axis represents the number of bits of information missing to uniquely identify the resource, while the vertical axis represents the number of requests with corresponding to particular levels of uncertainty.

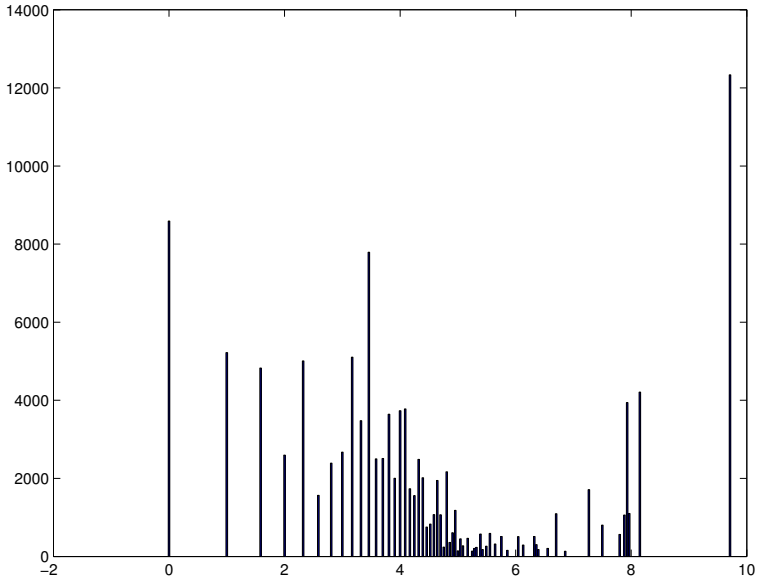

Fig. 1. Number of requests per bits of entropy missing for complete identification of the resource

The main problem with analyzing data from logs is that the only indication about the resource accessed is the URL. In many cases, such as when the page is dynamically created, or a standard page is displayed the URL is not a very good indicator of when a page is the "same" with another. This produced the peak that can be observed at 9.5 bits which contains 10% of the requests. It corresponds to the "Resource not found" page of the web server and for this reason it has been eliminated from the above statistics. The other large peaks represent other pages with the same content, and therefore the same length, but with different URLs. This is almost always due to the fact that the CGI [22] parameters in the URL are different.

The table and the graph presented here show that some information is provided through the length of the resource but in many cases this information is not enough to infer the exact page.

The logs also provide the request string for the resource. We could also use this information (quantized to 8 byte boundaries) to see if it can help us differentiate between resources. *ReqLen*, the probability a URL requested at a particular quantized length contains 3.07 bits of information. *ResLen*, the probability a reply has a particular length gives us 9.27 bits of information. Unfortunately the two are not independent, and their mutual information is 1.2 bits. Therefore the joint entropy of *ReqLen* and *ResLen* is 11.14 bits, which is the maximum amount of information that one can extract from these two distributions. This information should allow us to differentiate between $2^{11.14}$ resources but in prac-

tice resource lengths are going to start colliding after the site contains more than $2^{5.5}$ of them.

The above result shows that the length of the URL of the request contains some useful information, although because of the 8 byte boundaries of the TLS encryption it can only be used in conjunction with other techniques to differentiate between pages.

5 Traffic Analysis Using Multiple Requests

It has been highlighted in previous sections that the structure of a web site can provide an attacker with additional information about which pages are accessed, since users are more likely to follow this structure, using links, than by manually accessing random pages. In addition, some resources are automatically downloaded, such as images and HTML frames. Some experiments have been described that show how this structure could be used to find out which pages the user is accessing.

Instead of analyzing single request as in the previous experiments, we now consider a sequence of requests from a client to a particular site. These requests are encrypted, using TLS, so that only their approximate lengths are visible. The main question we would like to answer is: what sequence of accesses to pages gives us the most plausible explanation for the observed sequence of resource sizes. We try answering that question by modeling the site using a Hidden Markov Model.

5.1 The Hidden Markov Model

A Hidden Markov Model (HMM) is defined by the following:

S a set of hidden nodes $\{s_1, s_2, \ldots, s_N\}$.
K a set of symbols emitted $\{k_1, k_2, \ldots, k_M\}$.
Π where each element $\pi_i, i \in S$ of the vector represents the probability the process starts at node i.
A where each element $a_{ij}, i, j \in S$ of the matrix represents the probability there is a transition from node i to node j.
B where each element $b_{ik}, i \in S, k \in K$ of the matrix represents the probability node i emits a symbol k.

We define the hidden nodes S to be the resources present on the site we observe. The symbols emitted, K, are the sizes of retrieved documents the attacker can observe when looking at an encrypted link. Using standard HTTP logs we can calculate the probability matrices A, B and Π simply by counting the occurrences of each event.

It is important to note that each node could emit one out of many symbols, in the case of a dynamically generated page. The HMM can model this by having a symbol probability distribution for each node.

5.2 Building the Model

In order to construct the tables of the HMM the logs of the SRCF [20] web site were used. There are algorithms that create these tables just from sequences of observed data [21], but they are very expensive and less accurate than computing them directly.

The fact that, even anomymized, logs can be used to create such a model and extract information from encrypted requests should always be considered when logs are to be disclosed to third parties. It also poses an interesting legal problem in jurisdictions where it is not necessary to have a warrant in order to request the seizure of traffic data. In the light of these new ways of extracting information about the content requests the line between traffic data and actual message content is further blurred.

Web logs are not the only way the model could have been constructed. The structure of the static site is rich enough to construct such a model. Most of the structure in the real traffic results directly from resources that are linked together. These resources can either be requested automatically, such as images, or they can be requested manually, as it the case for hyper links. An HMM that models these static resources and the links between them still gives very accurate estimations of the hidden nodes corresponding to an encrypted communication.

Precise traffic data may not be available to the attacker because persistent HTTP connections are being used. In that case one cannot guarantee to observe the individual requests or replies and their sizes. The Hidden Markov Model could be adapted for this eventuality. The nodes could represent bundles of resources instead of single ones. Then the symbols emitted would not be the size of a single resource but all the possible lengths that the bundle could emit.

5.3 Finding the Most Plausible Path

When an attacker has extracted the above information from the logs, standard HMM algorithms can be used in order to find the most plausible set of hidden nodes that explain some observations. We use the Viterbi algorithm [21], which has a quadratic time complexity in the number of nodes.

In practice because of the constraints of HTTP traffic data we can optimize the Viterbi algorithm considerably. The inner loop of the algorithm considers one by one all the possible transitions from node to node. It is clear that because of the web's structure the transition graph is going to be extremely sparse. Therefore one needs only to consider the transitions that are observed. Furthermore we know that we are looking for a particular size and can therefore consider only the resources which emit such a symbol. So in practice the time complexity of the algorithm we implemented is proportional to the *Observed Sequence Length* $\times$ *The Average Number of Resources per Symbol* $\times$ *The Average Number of Transitions from any Node with a Particular Size to a Particular Node.* The optimizations we add also mean that if a particular sequence of observed symbols is not very ambiguous the time to find its corresponding hidden nodes is small. Care has been taken not to exclude any paths from the model. Even if

no instance of an event was observed a very small probability is still assigned to it. This is particularly useful for the transitions and the starting probabilities. The probabilities of symbols emitted are nearly always known. The algorithm was adapted from [21] in order to model symbols being emitted at the nodes, and optimized:

1. Initialization
   $\delta_j(1) = \pi_j b_{jo_1}, \forall j.\pi_j > 0 \wedge b_{jo_1} > 0$
2. Induction
   $\delta_j(t+1) = \max_{\forall i.a_{ij}>0} \delta_j(t) a_{ij} b_{jo_t}, \forall j.b_{jo_t} > 0 \wedge i \in \Psi(t)$
   $\Psi_j(t+1) = \arg\max_{\forall i.a_{ij}>0} \delta_j(t) a_{ij} b_{jo_t}, \forall j.b_{jo_t} > 0 \wedge i \in \Psi(t)$
3. Backtrack
   $\hat{X}_{T+1} = \arg\max \delta_i(T+1)$
   $\hat{X}_t = \hat{X}_{\Psi_{t+1}}(t+1)$

5.4 Results

Measurements were done using the HTTP log [20] data. Some logs were used to train the HMM, that then would try to classify which resources particular sequences of lengths are related to. In order to asses how well the HMM would perform if poorer training data was available two models were constructed:

The Rich Model is trained and recognizes resources using the complete information contained in the logs, including statistics on how popular particular pages are, and how popular particular links are.

The Poor Model drops all the information about the popularity of different pages, or links. It therefore represents the information an adversary with access only to the static site would have to construct a model.

The resources recognition rates for single resources versus sequences of seven resources can be summed by the following table, for the two models above:

Model	Single Resource	7 Resources
Rich	66.19%	88.72%
Poor	63.56%	86.23%

The fact that a majority of pages is recognized gives the attacker quite a good indication about which section of the site a user is browsing. Therefore even though the recognition rates will never be perfect enough information is extracted from the model to censor resources or perform surveillance.

5.5 Adapting and Improving the Model

The main drawback of using Hidden Markov Models is that the transition probabilities are independent from the previous requests. Therefore one cannot model "typical paths" that users take. One can overcome this by defining the nodes of the model as sets or lists of resources. In this case a transition exists between

one set and another if the user has already visited the previous resources and requests another one. Of course for many applications a complete history may not be necessary and a limited history could be enough. Instead of limiting the sets to the previous n resources one could only keep the "important resources" visited that give the most information about which other sites or pages are to be visited. Finally, one can construct sets using a mix of resources requested in the past and profiles that the users request match. That introduces a new need, to analyze the request patterns of users, and extract generic patterns that could help an attacker classifying request streams.

The techniques used to analyze resource size streams should take into account the fact that all the resources may not be requested. This can be due to browser settings or web caches. Fortunately, since our model was built using actual request logs it should be able to successfully cope with such cases, given that they have been seen before. On the other hand if the model is constructed, as suggested above, using the static structure of the web, one should take care to make it robust against unusual request patterns. In general the model should not be fooled by automatically downloaded resources being requested in a random order, not being requested or only a subset of them being requested.

6 Guidelines for Preserving Private Access

As it has been shown above using only TLS is not enough to protect the privacy of the users of Sam's service. On the other hand TLS could be used as a building block for more complex protocols that provide more effective privacy and anonymity. The aim is for the server to provide mechanisms to achieve these security properties, so that a client that only implements standard web technologies can benefit from them.

URL Padding The service should have fixed size URLs. This would limit the usefulness of the traffic analysis of the requests. These could be also generated randomly so that the length is different for each request of the same resource. In that case if only the first page of the service has a well known name and all users start from it, an attacker gains no additional information from the traffic analysis of requests. Additional information such as cookies returned could also be used to modulate the length of the URL.

Content Padding The content can be padded with junk of randomly distributed length. This will make the replies less useful for traffic analysis. There is always the fundamental problem that some messages might be more lengthy than others. In order to hide these they could be divided in two or more parts and linked together.

The padding could be added at the HTTP protocol layer, by using headers for padding. Cookies can also be used to provide padding and modulate the length of the resource (to a maximum of 4kb). For particular types of documents, such as HTML, the padding could be done by slightly changing the document. For example, HTML renderers do not differentiate between

one white space character or many. One could use this property to modulate the length of the document by adding white space characters.

Contribution Padding Most of the PUSH requests are created using a form provided by the server, in which padding data can be included. If the system is based on HTML forms, a HIDDEN field of variable length can be included in the form and the POST request to vary the length of the contribution. Special care should also be given to hide any timing correlations that might be used to link submissions with users.

Structure padding If both URL padding and content padding is implemented, structure padding should only be concerned with the observable automatic requests that an article initiates and also their quantity. By "observable" we mean things that give out information regarding the content of your request (Using this observable request the information available about which resource has been accessed has been increased). The quantity of traffic is also an important thing to have in mind: traffic padding in discrete chunks is only useful if the requests for these chunks are also distributed randomly, otherwise the frequency of the requests can act as a substitute to traffic analysis of the lengths themselves.

7 Conclusions

We have presented an overview of traffic analysis attacks that can be performed against protocols carried over TLS. Although the TLS protocol has been chosen to illustrate them, since it is the most widely deployed technology at the present time, most of the attacks can be used against other secure channels that do not provide any protection via padding or traffic flow control.

The techniques presented in the paper for analyzing simple requests and the models created for analyzing series of encrypted requests form a step toward a better understanding of traffic analysis attacks. We have seen in the literature plenty of techniques [23, 8, 24] that aim at protecting against such attacks but not many quantitative studies of the attack technologies themselves. It is very important if the community is serious about fielding systems resistant to snooping to have a deep understanding of how traffic analysis works and what its limits are. The time has come for the creation of a solid scientific understanding of these limits in the open community and quantitative research to be done on how existing or future systems providing unlinkability properties perform when they are under attack.

Acknowledgments. Removed for anonymous review.

References

1. Alan O. Freier, Philip Karlton, Paul C. Kocher, The SSL Protocol Version 3.0, November 18, 1996, `http://home.netscape.com/eng/ssl3/draft302.txt`.
2. T. Dierks, C. Allen, RFC2246: The TLS Protocol Version 1.0, January 1999.

3. R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, RFC2616: Hypertext Transfer Protocol – HTTP/1.1
4. T. Berners-Lee, R. Fielding, L. Masinter, RFC2396: Uniform Resource Identifiers (URI): Generic Syntax, August 1998
5. SafeWeb, One Click Privacy Anywhere, Anytime. `https://www.safeweb.com/`.
6. The Anonymizer `http://www.anonymizer.com`
7. Jean-Franois Raymond, Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems, H. Federrath (Ed.), Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000, Proceedings, LNCS 2009, p. 10 ff.
8. Lance Cottrell, MixMaster Resource centre, `http://www.obscura.com/~loki/`
9. Oliver Berthold, Hannes Federrath, and Stefan Kopsell, Web MIXes: A System for Anonymous and Unobservable Internet Access, H. Federrath (Ed.), Designing Privacy Enhancing Technologies, International Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, USA, July 2000, Proceedings, LNCS 2009, p. 115 ff.
10. D. Song, D. Wagner, and X. Tian. Timing Analysis of Keystrokes and SSH Timing Attacks In 10th USENIX Security Symposium, 2001.
11. Edward W. Felten and Michael A. Schneider. Timming Attacks on Web Privacy. Proc. of 7th ACM Conference on Computer and Communications Security, Nov. 2000.
12. Paul Syverson, Gene Tsudik, Michael Reed, and Carl Landwehr. Towards an Analysis of Onion Routing Security. LNCS 2009, p. 96 ff.
13. Andrew Hintz, Fingerprinting Websites Using Traffic Analysis, Privacy Enhancing Technologies Workshop 2002, San Francisco.
14. Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Wilf Russell, Venkat Padmanabhan, Lili Qiu, Statistical Identification of Encrypted Web Browsing Traffic, 2002 IEEE Symposium on Security and Privacy, Oakland.
15. M. Levene and G. Loizou. Computing the entropy of user navigation in the web. Research Note RN/99/42, Department of Computer Science, University College London, 1999.
16. E. Rescorla, RFC 2818: HTTP Over TLS, May 2000
17. R. Khare, S. Lawrence, RFC2817: Upgrading to TLS Within HTTP/1.1, May 2000
18. Independent media centre, `http://www.indymedia.org:8081/fbi/`
19. BBC News, `http://news.bbc.co.uk/`
20. The Student Run Computing Facility, `http://www.srcf.ucam.org/`
21. C. Manning, H. Schutze, Foundations of Statistical Natural Language Processing, The MIT Press, ISBN 0-262-13360-1.
22. The Common Gateway Interface (CGI) Specifications `http://www.w3.org/CGI/`.
23. The Java Anonymous Proxy `http://anon.inf.tu-dresden.de/`.
24. D.Chaum, (1981), Untraceable electronic mail, return addresses, and digital pseudonyms, Communications of the ACM, Vol. 24, No. 2, February, pp. 84–88.