Statistical Identification of Encrypted Web Browsing Traffic

Qixiang Sun
Daniel R. Simon
Yi-Min Wang
Wilf Russell
Venkata N. Padmanabhan
Lili Qiu

March 8, 2002

Technical Report
MSR-TR-2002-23

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

To appear in *Proc. of IEEE Symposium on Security and Privacy*, May 2002.

Statistical Identification of Encrypted Web Browsing Traffic

Qixiang Sun[1] Daniel R. Simon[2] Yi-Min Wang[2] Wilf Russell[2]
Venkata N. Padmanabhan[2] Lili Qiu[2]

Abstract

Encryption is often proposed as a tool for protecting the privacy of World Wide Web browsing. However, encryption–particularly as typically implemented in, or in concert with popular Web browsers–does not hide all information about the encrypted plaintext. Specifically, HTTP object count and sizes are often revealed (or at least incompletely concealed). We investigate the identifiability of World Wide Web traffic based on this unconcealed information in a large sample of Web pages, and show that it suffices to identify a significant fraction of them quite reliably. We also suggest some possible countermeasures against the exposure of this kind of information and experimentally evaluate their effectiveness.

1 Introduction

The rise of the World Wide Web has triggered serious concerns about the possible threats to privacy associated with Web browsing. These include possible inadvertent disclosure of location or other personal information through data traffic that reveals the browsing user's identity and/or associates him or her with browsing a particular Web page. Even partial revelation of such information can cause embarrassment, financial loss, or even physical harm.

For example, a user may reveal, simply by accessing a private home Web server from abroad, that the resident is out of town (and therefore that the home is vulnerable to burglary). The observer need only notice that the home Web server is being accessed, and that the originating IP address of the HTTP request is not in the same locale as the home/server–usually an easy thing to determine [22]. The inference can thus be made that the resident (the most likely–perhaps the only likely–browsing user of the private Web server) will not soon return home.

In other cases, the user can actually be identified, either directly, through explicit identifying information in the traffic, or indirectly, through inference–say, by noticing a common IP address of origin with other identifying traffic. In those cases, a user found to be browsing Web pages containing certain types of medical or financial information may inadvertently reveal, through implied interest in that information, embarrassing or financially confidential information about him- or herself.

For these reasons, considerable research has been directed at techniques for "anonymizing" Web browsing–that is, hiding the connection between a particular user and the Web pages he or she is accessing. Most attention has focused on two main tools: data encryption, to hide information that might reveal either the user's identity or the Web page's, and one or more intermediate proxies, to hide from any particular proxy (or from an observer of network traffic) the connection between the browsing user's network address and the Web site's [1, 4, 14, 25].

Even when multiple proxies are used, however, the first link—between the user and the first proxy—is the most vulnerable to attack, since the attacker (whether the first proxy itself, the user's ISP, or perhaps an eavesdropper—say, on a wireless link) can immediately determine the user's network address. To foil such attacks, encryption is essential; moreover, it must genuinely render the Web content being browsed unidentifiable. The question thus arises: how effectively does encryption of Web traffic hide its source?

To answer this question, we conducted a large study of roughly 100,000 Web pages, attempting to distinguish them based solely on information that might well be available even if they were being accessed through an encrypted channel—say, one protected by SSL [13] or its successor, TLS [10]. This "traffic signature" information includes, essentially, the number of objects requested as part of a Web page download, and the

[1]Department of Computer Science, Stanford University, Stanford, CA 94305 USA. Email: qsun@cs.stanford.edu. This work was done while interning at Microsoft Research.

[2]Microsoft Research, Redmond, WA 98052 USA. Emails: {dansimon, ymwang, wilfr, padmanab,liliq}@microsoft.com

lengths of those objects (subject to various possible padding schemes). We found that without extremely aggressive length padding, a relatively straightforward identification algorithm could in fact accurately identify many Web pages within the study with very low false positive rates based on number and sizes of objects alone. The algorithm was efficient and easily automated, and required no sophisticated manual analysis. Naturally, the number of false positives might be expected to increase if applied to the entire Internet, but such an effective initial "pruning" of possibilities makes the likelihood of effectiveness of more sophisticated methods much higher.

We also explored some alternative countermeasures, including unconventional types of padding and more elaborate length-hiding mechanisms, including some which are noticeably more effective than standard padding techniques.

2 The Experiment

2.1 Setting

When a typical browser fetches a Web page, it issues an HTTP "GET" request to the address indicated by the page's URL, and receives in response an HTML "object" which may in turn contain references to other Web objects. These objects are then fetched in turn, synchronously (although in parallel on multiple TCP connections, so as to speed the process and prevent a single failed "GET" from delaying the downloading of the rest of the page). Thus a given Web page results ultimately in the downloading of a certain (fixed or variable) number of objects in a (possibly variable) order. Each of these in turn may have either fixed or variable length.

We assume that both the outgoing HTTP requests and returned objects are strongly encrypted, revealing no identifying information, and moreover that the IP address of the recipient of/responder to the request is merely a proxy server whose identity provides no information about the real source of the Web page. For example, the browser may have established an SSL connection to the proxy server, and be forwarding (encrypted) HTTP requests to the proxy over that connection. In this case, the requests are protected by the encryption on the SSL channel, but the sizes of the returned objects are clearly discernible from the synchronous "GET" requests (to within the cipher's block size, if a block cipher is used). (As we will discuss in Section 4.3, avoiding the synchronous "GET" requests by using HTTP pipelining [12] can hide the object sizes. Unfortunately, current browsers do not use HTTP pipelining.) Alternatively, the HTTP requests generated by the browser, and the corresponding responses received by it, may be intercepted by a more sophisticated intermediate layer on the local machine which implements its own encryption protocol, possibly including large amounts of data "padding" (particularly of eturned HTTP objects) to disguise their true lengths.

Normally, a browser caches recently-fetched objects, to speed the presentation of pages containing these objects. However, it has been shown in [11] that in this case any server, by including a request for an object from a particular site in its Web page (say, between requests for two objects from the server itself), and measuring the delay introduced by this fetch, can determine with high fidelity whether the included object was cached, and thus whether the browsing user had previously visited that site in a single-user system setting (such as a home desktop machine). Hence we assume that object caching has been disabled, to guard against these timing attacks by servers.

We further assume that an adversary is monitoring encrypted traffic, searching for examples of access to one of a set of Web pages. For example, the adversary may be searching for pages from Websites of a sensitive nature, or those that implicitly reveal information about the browsing user. The adversary can presumably maintain an up-to-date database of object-number and object-length profiles of the pages of interest. The adversary may not need to identify an example of access to a particular "interesting" page with 100% accuracy; however, too much "noise" among detected access instances would render the observations useless to anyone trying to exploit them.

Figure 1 describes in more detail the hypothetical adversary architecture on which we based our experiment. The adversary compiles traffic information on particular pages of interest, collects traffic from potential viewers of those pages, and evaluates the similarity of the traffic patterns to determine if a particular viewer is viewing one of the "target pages". Although contextual information (such as the viewer's past history of traffic) may be used, our experiment only compares the traffic patterns themselves.

Depending on the adversary, different rates of false positives and false negatives may be acceptable. In most scenarios, however, the value of monitoring data will be most severely affected by the false positive rate, for several reasons:

- Browsing users tend to revisit sites multiple times, fetching multiple pages, and usually one identification of a targeted access is enough; hence high

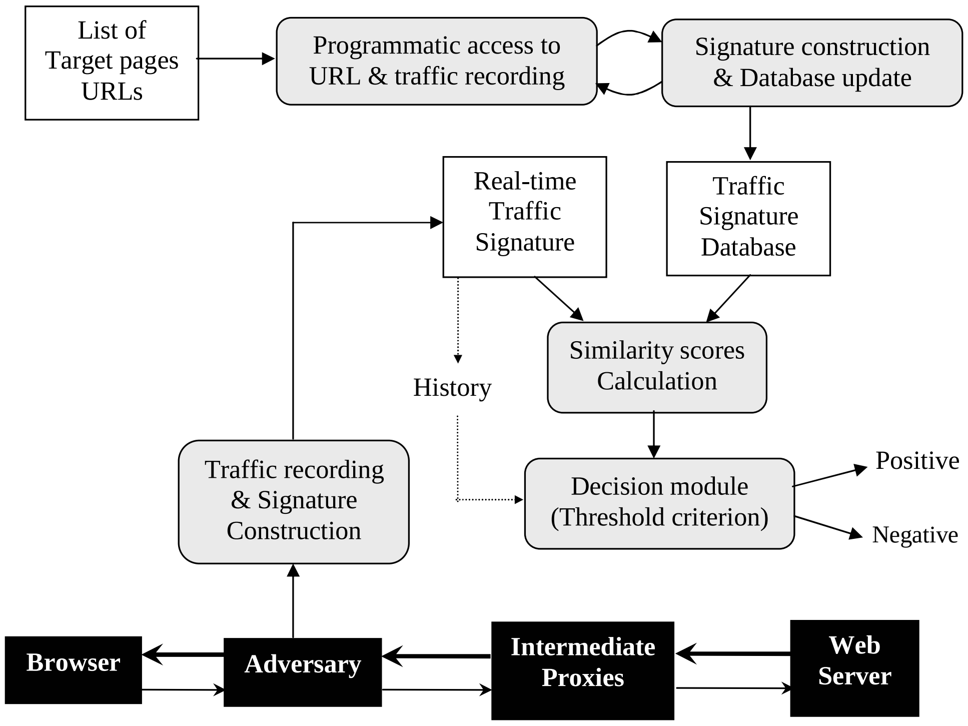
List of Target pages URLs
Programmatic access to URL & traffic recording
Signature construction & Database update
Real-time Traffic Signature
Traffic Signature Database
Similarity scores Calculation
History
Decision module (Threshold criterion)
Positive
Negative
Traffic recording & Signature Construction
Browser
Adversary
Intermediate Proxies
Web Server

Figure 1. The adversary architecture.

false negative rates are not that much of an obstacle.

- Since the overwhelming majority of browsing traffic is likely to be uninteresting, even a fairly small false positive rate can result in huge numbers of false positives in absolute terms.
- Since false positives are unlikely to disappear entirely, positive reports may require significant extra analysis to verify (such as examining contextual information to determine the plausibility of the detected traffic). Hence reducing this overhead is a high priority.

Thus we assume the adversary to be attempting to identify Web pages with as low a false positive rate as possible, while still achieving a significant rate of true positive identifications.

2.2 Procedure

We collected traffic signature information on a sample of just under 100,000 Web pages, from a wide range of different sites. The pages were obtained from the DMOZ Open Directory Project link database (http://dmoz.org), half of them chosen from various categories of "sensitive" sites to which an adversary might be interested in spotting visitors, and the other half chosen randomly. The information we examined simply consisted of the number and sizes of the (unordered) set of objects fetched by a browser (Microsoft Internet Explorer version 5.5) accessing that page. The objects' number and sizes were determined solely by observing the chunks of response data (blocks of packets) received by the browser between blocks of request packets emanating from the browser in a trace of the browser's TCP connections. Thus no information was used that would have been obscured had the data passed across the connections been encrypted.

A small subset of just over 2000 "target pages" in the sample (from two particular subcategories of the "medical information" category), were also visited in advance, to collect a "signature database", before visiting the entire sample (including non-target pages). We then chose a simple scalar "closeness" metric $Sim(s1, s2)$ for measuring the similarity between two signatures, to be used to determine how well each given signature matched one of those in the sample. Viewing the pages as multisets of object lengths, we chose Jaccard's coefficient ($Sim(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$) as our metric [28], using the standard definitions of multiset intersection and union—minimum number of repetitions for intersection, maximum for union [16].

To evaluate the success of our hypothetical adversary, we defined the following categories for pages in our sample and "target" subsample:

1. **Identifiable page**: given a set T of target pages and a page $t \in T$ (identified as t' when fetched a second time), t is an **identifiable page with respect to T** if (1) $\forall\, u \in T$ and $u \neq t, Sim(t', t) > Sim(t', u)$ and (2) $Sim(t', t) \geq c$, where c is the similarity threshold for Sim.

   That is, an identifiable page in the subsample is one that our metric correctly matches with its incarnation in the subsample when revisited. This definition excludes pages that either are too similar to any others in the same subsample or vary too much in traffic signature between different accesses. If $S \subset T$ is the set of identifiable pages in the subsample, then the **identification rate of T** is $\frac{|S|}{|T|} \cdot 100\%$.

2. **Potential false positive**: given a target page t and a non-target page n, n is a **potential false positive of t** if $Sim(n, t) \geq c$, where c is the similarity threshold for Sim.

   Put simply, a potential false positive is a non-target page that our similarity measure and threshold scheme might possibly identify as a target page. Note that the definition of a potential false positive is independent of the rest of the subsample. It allows us to compute an upper bound on the number of actual false positives without considering all possible sets of target pages. The actual number of false positives is smaller, as the following definition explains:

3. **Actual false positive**: given a subsample of target pages T and a non-target page $n \notin T$ in the larger sample, n is an **actual false positive with respect to T** if (1) $\exists\; t \in T$ for which n is a potential false positive of t, and (2) $Sim(n, t) > Sim(n, u)$ for all $u \in T$ and $u \neq t$.

   In other words, a potential false positive n will not result in an actual false-positive decision if n is a potential false positive for more than one target page and the similarity scores are tied (since a detection algorithm tuned to minimize false positives will refuse to identify n as either one of the two plausible candidate target pages). For the set N of non-target pages, if $F \subset N$ is the set of actual false positives with respect to T, then the **actual false positive rate of N with respect to T** is $\frac{|F|}{|N|} \cdot 100\%$.

4. **K-identifiable page**: given a set of target pages T and a set of non-target pages N, a page $t \in T$ is a **K-identifiable page with respect to T and N** if (1) t is an identifiable page with respect to T, and (2) for $P \subset N$ consisting of all $n \in N$ that are potential false positives of t, $|P| \leq K$.

   That is, a K-identifiable page is an identifiable page from the subsample that generates at most K potential false positives in the overall sample. Thus, a uniquely identifiable page is "0-identifiable", generating no false positives in the overall sample. Given T and N, if $S \subset T$ is the set of K-identifiable pages in the subsample, then the **K-identifiability rate of T with respect to N** is $\frac{|S|}{|T|} \cdot 100\%$.

For the reasons explained earlier, our goal was to determine if a threshold exists that allows a significant fraction of target pages to be identified while maintaining very low false positive rates. Naturally, the unique identification rate would depend on such factors as the amount and method of padding used in the encryption, and the variability of the pages; determining how much each of these factors affected the unique identification rate was also a goal.

3 Results

For our sample of 100,000 pages, the median number of objects in a page was 11. After rounding object sizes to the nearest 32 bytes (to take into account minor HTTP header changes in responses from the same page), the entropy of the object size distribution in the sample was about 8.4 bits. Thus a typical Web page's traffic pattern (ignoring object retrieval order) can be viewed as an unordered set of 11 objects, yielding a total of $8.4 \cdot 11 - \log_2(11!) > 67$ bits of information—certainly more than enough (in theory) to recognize individual Web pages from the entire range available on the World Wide Web. (According to [2], the Google search engine fully indexed roughly 1.5 billion pages as of December, 2001, which would require only 31 bits to distinguish them.)

Figure 2 shows the identification rate of the 2191 target pages and the actual false positives rate of the 98496 nontarget pages as a function of the threshold used in the similarity metric. It is clear that for a substantial intermediate range of threshold values, a high identification rate coincides with an very low false positive rate. In particular, a threshold of 0.7 gives an identification rate of about 75%, and a false positive rate of less than 1.5%; that is, fewer than 1.5% of pages outside the target set in our larger sample were incorrectly identified as a target page from our subsample.

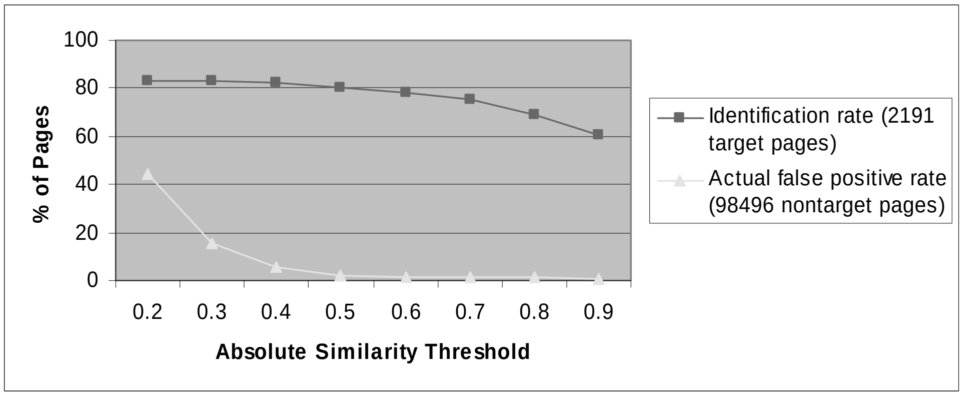

Figure 2. Identification rate and actual false positive rate as a function of the similarity threshold (2191 target pages vs. 98496 non-target pages).

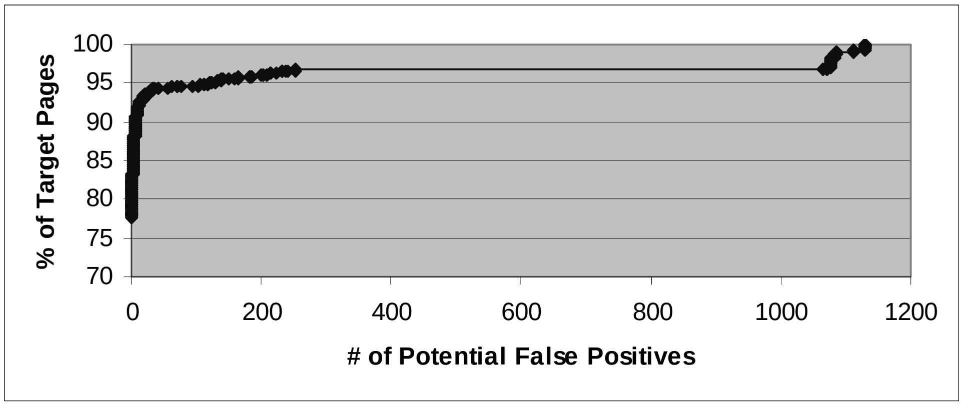

Figure 3. CDF of target pages with potential false positives.

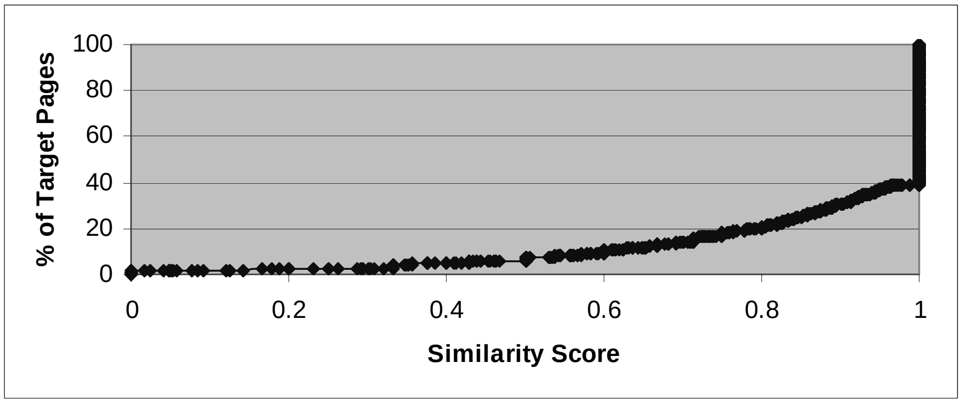

Figure 4. CDF of similarity scores for two consecutive accesses of the same page.

Low as this false positive rate is, it masks the disproportionately large effect of a small subset of the target pages. As Figure 3 shows, a significant fraction of target pages that generate potential false positives in this case generate a large number of them; these "generic-looking" pages appear to generate the bulk of potential false positives. (Many of these may not generate actual false positives; that is, since they may match multiple target pages equally, they could easily be discarded as unlikely true positives. For example, various types of error messages, which tend to fall into a few standard classes, produced the sharp rise on the top right corner of Figure 3's CDF graph. These are not necessarily easily distinguishable from normal pages; they may, for instance, be generated and formatted by the Web server and returned as normal pages.) Meanwhile, over 77% of pages did not even generate a single false positive from the overall sample. This latter statistic is thus the more relevant measure of the feasibility of identifying Web traffic than the fraction of the overall sample that would be a false positive for some member of the target set, since an attacker, by omitting generic-looking pages, could expect to identify the remaining target pages with an extremely low false positive rate.

On the other hand, some pages—regardless of the number of false positives they generate—are in practical terms non-identifiable by virtue of their highly dynamic nature. Figure 4 shows how pages vary between accesses. Just under 40% do not match exactly even when re-accessed immediately, and about 14% do not even achieve a similarity score of 0.7. In a smaller sample of 20,000 pages not shown here, we found that the difference between immediate and day-to-day changes is fairly small, indicating that pages can be roughly categorized as stable or variable between accesses, without regard to the exact time interval between the accesses. (Since the target set is assumed to be relatively small, the adversary can be assumed able to update its signature data at least daily.)

Henceforth, we will concentrate on rates of unique identifiability (defined as meeting the similarity threshold, with no potential false positives) of our target set against the entire 100,000-page sample as our chief measure of the success of our identification method. We will examine how various countermeasures affect this unique identifiability rate, as well as the k-identifiability rate for very small values of k. These figures hopefully give some indication as to the "noise" levels that an adversary would face when trying to detect encrypted browsing of a chosen target set of Web pages.

4 Countermeasures

We next describe several traffic-shaping mechanisms that can be used to make the attackers' job much harder. All of them require varying degrees of additional effort by the Web server and/or client software to protect the anonymity of clients. They can be classified into three categories: padding, mimicking, and morphing.

4.1 Padding

Padding is often cited in the literature as a means to disguise traffic volume. Typically it is used to create uniform-looking blocks of data out of blocks of varying (and thus distinguishable) sizes. One implementation of "onion routing", for instance [27], transmits 128-byte blocks of data, padding blocks if necessary to prevent blocks from being distinguishable based on length. In the case of transmissions larger than the block size, the only information revealed about the length is the nearest larger multiple of the block size.

Although padding schemes can help, they are far less effective than one might assume, particularly in cases such as Web browsing traffic, where a collection of associated padded lengths can be analyzed. Figure 5 shows how a standard linear padding scheme reduces the fraction of uniquely identifiable pages in our subsample; in each case we varied the similarity threshold to optimize the fraction of uniquely identifiable pages, as well as the fraction of "nearly uniquely identifiable" pages (1- and 2-identifiable pages). A 128-byte scheme was very weak; over half the pages in our subsample remained uniquely identifiable. Rounding object sizes up to multiples of 4 Kbytes (effectively nearly doubling transmission overhead, based on a median object size of 2.5Kbytes) still allowed unique identification of nearly 18% of pages. In both cases, a further 8% of pages were nearly uniquely identifiable. It took a minimum object size of between 8 and 16 Kbytes to reduce the fraction of uniquely identifiable pages below 5%.

Of course, a complete absence of false positives in our sample does not imply that there are no false positives in the World Wide Web as a whole. On the other hand, we do not consider a complete absence of false positives to be necessary for effective identification to occur; we assume that an adversary can apply more careful (and expensive) scrutiny of traffic to further eliminate false positives, as long as an efficient automated screening method exists to allow the extra processing to be limited to a relatively tiny fraction of observed traffic. (Such processing might involve correlation with various kinds of contextual informa-

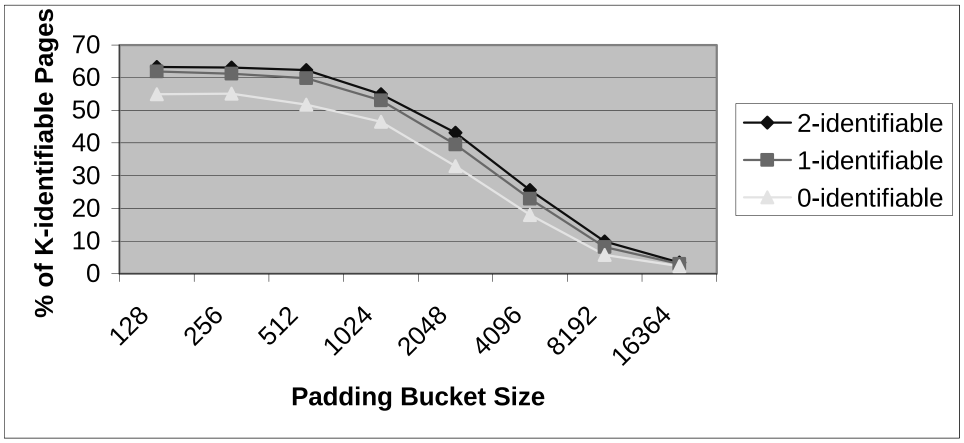

Figure 5. ***K*-identifiability rate with linear padding of various bucket sizes.**

tion, such as the signatures of prior and subsequent pages, the browser's IP address, time of day, and so on.) It appears that a significant number of pages are indeed amenable to such screening, with very low (if ultimately non-zero) false positive rates

An alternative method we call "exponential" padding, in which objects are padded out to number of bytes which is a power of 2 (rather than a multiple of a fixed size), works somewhat better than the standard ("linear") method. As Figure 6 shows, exponential padding with a 128-byte minimum block size reduces the unique identifiability rate to 27%, with a further 9.5% nearly uniquely identifiable. A 5% unique identifiability rate requires somewhere between a 4- and 8-Kilobyte minimum block size. Of course, exponential padding can be much more expensive than linear padding, since even large objects can be as much as doubled in length.

Another possible padding method is to add extraneous objects of arbitrary length to the page. This technique has only modest success against our similarity measure. As Figure 7 shows, the presence of randomly-sized extraneous objects depresses similarity scores for true positives, requiring our decision module to accept (and thus to be able to distinguish from others) pages with relatively low similarity scores. Figure 7 also shows that if the threshold is too small (e.g. 0.3 instead of 0.4), the identifiable rate decreases because of more potential false positives. However, even when we reduced our similarity threshold to take this effect into account, our measure still generated no false positives for over 40% of target pages, and one or two false positives for a further 8.5%, when extraneous objects of random size up to 10 Kbytes were added to bring the total number of objects to a multiple of 10. Increasing the padding factor didn't help that much, as shown in Figure 8; when pages were padded to a multiple of 20 objects, the unique identifiability rate was still over 30%. However, additional experiments showed that combining extraneous objects (to a multiple of 15 objects) with aggressive object padding (to a multiple of 2 Kbytes) was quite effective; even using the optimal similarity threshold, only 3.8% of doubly-padded pages were uniquely identifiable, with a further 2% generating one or two false positives. Again, though, such a combined padding scheme can be quite inefficient; a "median page" of 11 objects each of length 2.5 Kbytes would be nearly tripled in total length.

4.2 Mimicking

Another approach is to find patterns of Web traffic that are common to many different Web pages, and try to tailor one's Web pages in order to hide among them. For example, popular Web hosting services (like Yahoo, Angelfire, etc.) often provide standard templates for Web pages whose profiles can be mimicked. Alternatively, a Web page's content could be tailored to mimic particular widely accessed pages, in order to guarantee numerous false positives. Home pages of particular popular Web sites are obvious candidates.

4.3 Morphing

The third approach tries to make the traffic patterns generated by client accesses different from those expected by the attackers. There are at least six simple methods to accomplish this:

1. HTTP 1.1 byte-range requests [12] can be used by the client to randomly break one Web page into multiple chunks with potentially overlapping

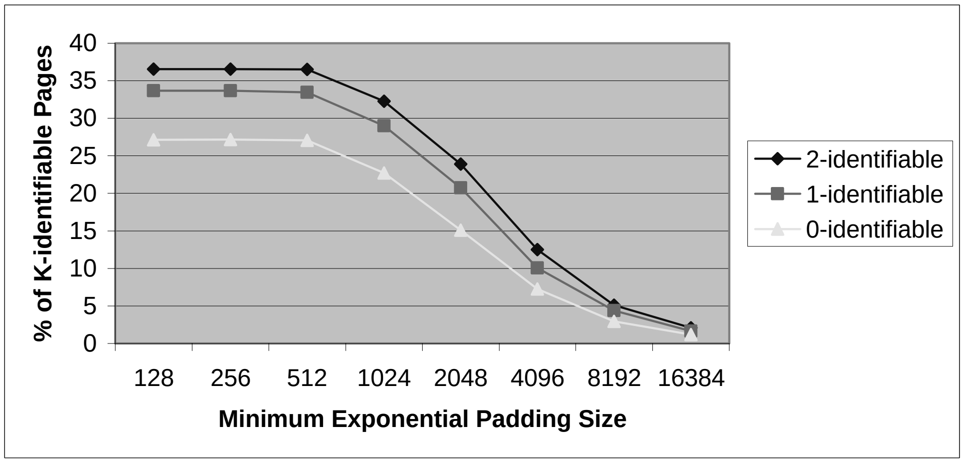

Figure 6. K**-identifiability rate with exponential padding of various sizes.**

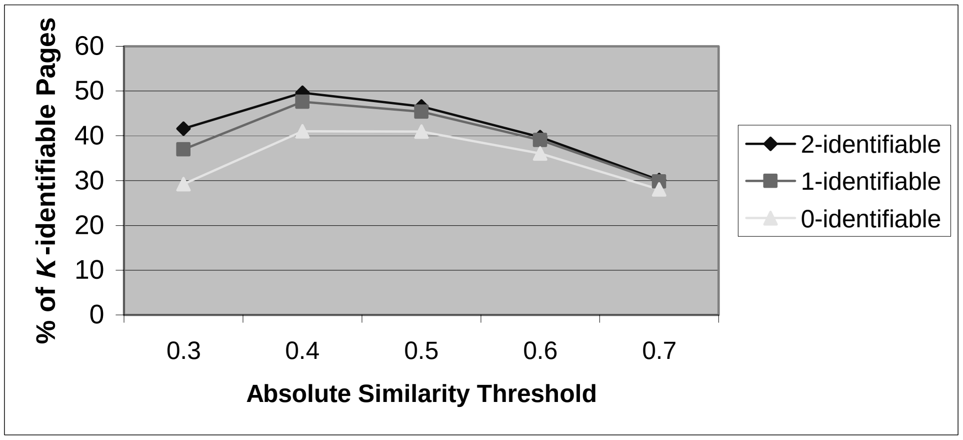

Figure 7. K**-identifiability rate for padding with random objects (total number of objects padded to a multiple of 10).**

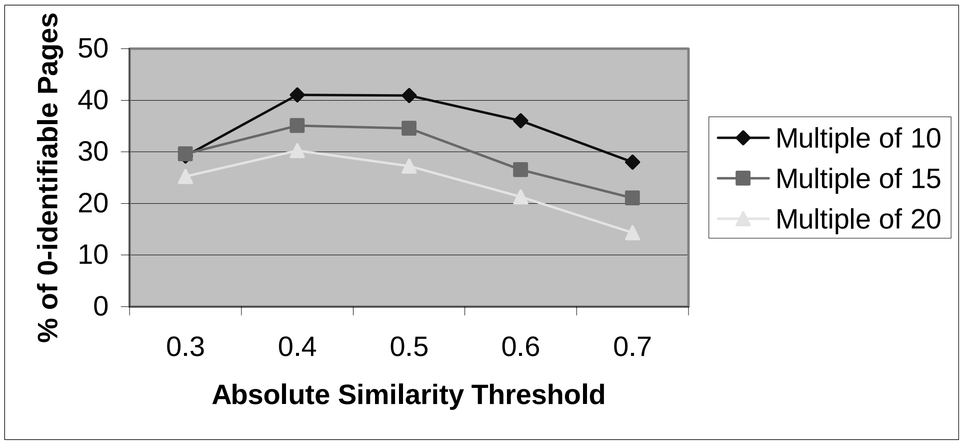

Figure 8. 0-identifiability rate for different random object padding (to multiples of 10, 15, and 20).

ranges. Thus object sizes are effectively completely disguised, and only a (padded) total size of all objects is revealed.

2. The HTTP content negotiation mechanism, which allows clients and servers to negotiate the format/type of some objects when applicable, can be used to alter traffic patterns. The multiple formats may correspond to encodings of different quality (for images), different languages (for text), etc. Content negotiation can be either server-driven (with the server deciding which format to send, with or without input from the client) or agent-driven (with the server telling the client about the formats available and the client picking one).

3. HTTP pipelining [12] can be implemented to allow multiple objects to be requested and returned together, so that the boundaries between them (and thus their sizes) are obscured if they are encrypted. After first downloading and parsing the HTML file for a Web page, the client can then issue a pipelined sequence of requests for the embedded objects, if all the objects are from the same Web server. The transfer would thus typically happen in two chunks—one corresponding to the HTML and the other corresponding to all of the embedded objects as a group. However, "one-chunk transfer" is possible if a (trusted) proxy first downloads all of the files and then sends them to the client in one chunk.

4. Prefetching/pushing of objects belonging to a particular page that a user will most likely visit effectively removes those objects from the traffic pattern associated with that page. Prefetching/pushing can be either client-based, with a client browser issuing requests for objects that are not directly requested by the user, or server based, with a Web server proactively pushing content to clients.

5. A Web ad blocker [3] can be extended to randomly block a (possibly varying) subset of the objects that are advertisements. The blocker uses a customized name resolution file to block name resolution for certain sites, thus preventing the browser from issuing HTTP GET requests for objects on those sites. The effect would likely be similar to that of a limited amount of random-object padding, as described above.

6. Finally, the user can run multiple browser instances simultaneously, each visiting a different page, so that objects from multiple Web pages interleave with one another.

Our experiments show that the pipelined "two-chunk" delivery scheme implementable under HTTP/1.1 is not overwhelmingly effective, allowing a 36% unique identifiability rate for our target set and sample size. (Interestingly, for our sample collection of 100,000 pages, two-chunk delivery yields about 16.8 bits of information, or 110,000 possibilities. Therefore the theoretical limit of the unique identifiable rate, using the solution of the "Coupon Collectors Problem" [21], is roughly $\frac{1}{e} \approx 37\%$. Since our result of 36% is very close to the theoretical limit of 37%, it appears the distribution of page lengths in our sample collection is indeed random, and harvesting the full entropy of the available length information is quite easy in practice.) In this experiment, we assumed all embedded objects are from the same Web server, thus possible to perform "two-chunk" delivery. In reality, usually more than two chunks of data are fetched because a page may contain objects from multiple sites which must be fetched in separate chunks; therefore in practice, the unique identifiability rate will be higher than the 36% observed in our experiment.

On the other hand, techniques that reveal only total page size suffice to make page identification extremely difficult. As Figure 9 shows, collapsing Web pages in our target set into a single object (with a single size) reduces the unique identifiability rate to 7%, with a further 3% of sites nearly uniquely identifiable. The flatness of the curve as k increases also suggests that 90% of the target pages are inherently unidentifiable by only using the total page size. Moreover, using a 256-byte padding scheme on the total page size results in almost every page having at least one potential false positive, with over 97% having at least ten of them. "One-chunk" pipelining and byte-range requests can thus both be expected to achieve this level of success.

4.4 Countermeasure Costs

The above countermeasures are all associated with extra costs which may make them prohibitively expensive or inconvenient to implement. Table 1 summarizes the costs of the various countermeasures, indicating whether special support is needed either in the client (browser), the server, or the content generation process. Some techniques, as indicated, require only features that are part of the HTTP/1.1 standard, but are often not fully implemented in popular client or server software. (For example, byte-range requests require both client and server support, but the necessary server support consists only of full implementation of the feature according to the HTTP/1.1 standard, whereas the client requires implementation of the

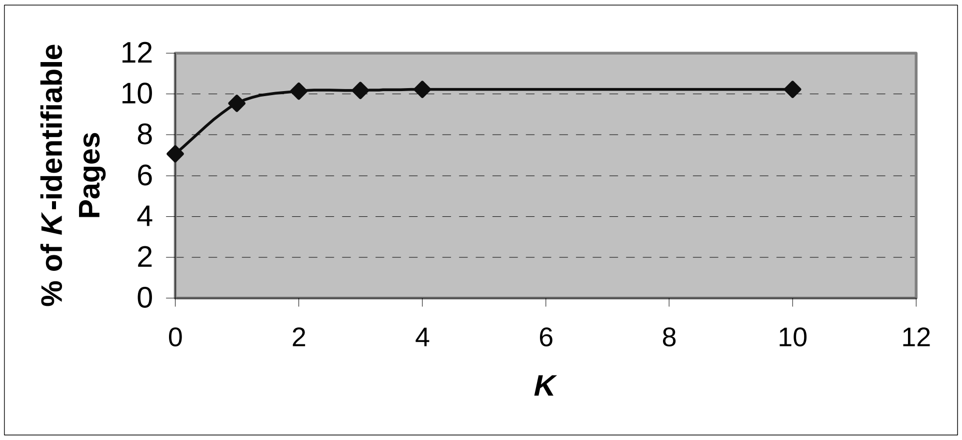

Figure 9. K-identifiability rate as a function of K under "one-chunk" delivery.

Technique	Client support?	Server support?	Content support?	Other Cost/Difficulty
Padding (object size)	Depends on object type	No	Yes	Wasteful data transfer
Padding (hidden objects)	No	No	Yes	Wasteful data transfer
Mimicking	No	No	Yes	May be hard to do (or maintain as mimicked page changes)
Byte-range requests	Yes	Yes (HTTP/1.1)	No	Overlapping requests, if any, cause wasteful data transfer
Client-based Prefetching	Yes	No	No	Prediction may not be perfect, resulting in wasteful data transfer
Server-based pushing	Yes	Yes	No	Prediction may not be perfect, resulting in wasteful data transfer
Content negotiation (server-driven)	No	Yes (HTTP/1.1)	Yes	Wasteful data transfer or loss in quality; extra server storage
Content negotiation (agent-driven)	Yes	Yes (HTTP/1.1)	Yes	Wasteful data transfer or loss in quality; extra server storage; extra network round-trip for the initial server format advertisement [17]
Web ad blockers	Yes	No	No	Customized name resolution file and loss of some objects
Pipelining (1-chunk)	Yes	Yes	No	Need trusted proxy to assemble the entire page on behalf of the client
Pipelining (2-chunk)	Yes (HTTP/1.1)	Yes (HTTP/1.1)	No	None
Multiple browsers	No	No	No	Hard to coordinate; wasteful data transfer

Table 1. Comparison of Various Countermeasures.

request-generation mechanism. Byte range requests also do not require special content manipulation, unlike, say, mimicking, which requires content to be tailored to match the signature of some other site.) Also listed are other miscellaneous costs or difficulties in employing each technique, the most common of which is the redundant traffic often generated as a result of the countermeasure.

For example, padding of some object sizes may require only content adjustment (such as adding "filler" comments to HTML objects), while some object types may require client cooperation to remove the padding. Hidden objects, on the other hand, can always be added by the server without any special client support. In either case, though, extra bandwidth must be used for its transmission.

The effectiveness and the overhead of each technique in hiding web page identities depend heavily on the parameters used for the technique. For example, padding objects to a multiple of 16 Kbytes can be as effective as one-chunk pipelining without any padding. However, our experiences indicate techniques that reduce the number of perceivable objects per page are more effective, easier to deploy, and waste less bandwidth.

5 Related Work

General anonymization of network traffic goes back to Chaum's "mix" idea [6]; a practical descendant is "onion routing" [14]. Other theoretical treatments include [7] and [23]; more practical work, geared specifically towards Web traffic, includes the LWPA [15], CROWDS [25] and HORDES [26]. The mix-based and theoretical approaches all use combinations of some form of encryption and interposition of intermediaries, and impose a rigid structure on the "shape" of traffic (synchronized transfer of standard-length blocks of data, and even dummy "covering traffic") to prevent leakage of information through timing or size of messages. More practical onion routing dispenses with measures such as synchronization and covering traffic, and is thus vulnerable to the kind of traffic analysis described here.

The Web-oriented approaches tend to downplay or eliminate the role of encryption, assuming an adversary with limited traffic observation powers, and relying instead on techniques such as trusted (or multiple random) intermediaries, pseudonyms and multicast to disguise the identities of browsing users. LPWA focuses on protecting browsers from revealing user identity to servers, as opposed to revealing Web traffic to eavesdroppers, and hence doesn't use encryption at all. CROWDS uses encryption between browser and proxy, to foil eavesdroppers, but does not attempt end-to-end encryption. In [25] it is also recommended that proxies fetch all objects in a page from the server before passing them on, to prevent other proxies from distinguishing end-user browsers from proxies by timing object requests. (This would be equivalent to our notion of "one-chunk delivery".) There are also several actual commercial anonymity services, such as the Anonymizer [1] and Zero Knowledge Systems' Freedom service [4]; these use encryption combined with proxies in ways that may be vulnerable to our statistical attacks.

The use of traffic attributes like data length to identify public content is not new, of course; in fact the CDDB music recognition service [18] uses such information to recognize music CDs. Observing the volume and timing of encrypted traffic has also been discussed [24, 27] as a method for attacking mix-based systems, in spite of the techniques mentioned above. Timing attacks on cached Web data have also been proposed [11] as a way to track Web browsing behavior. And of course, standard browser features such as cookies [20] can by themselves defeat anonymization.

The specific vulnerability of encrypted Web traffic to analysis was briefly examined in [8] where each web page is modeled as two numbers—the HTTP request size and the total page size. Their primary goal is to identify individual pages within a single Web site using the internal link structure. In contrast, our work uses individual object sizes in a page to identify individual pages from the entire Internet. The technique used in [8] can easily be adapted to our adversary architecture as the past history component in Figure 1. A slightly different problem of identifying individual objects within a Web page was also studied in [9].

6 Summary and Future Work

We have shown that even encrypted, padded Web traffic can yield considerable information about the source of its contents. Moreover, the revealed information appears sufficient to be useful to an attacker monitoring such encrypted traffic in an effort to undermine the privacy of Web browsing.

One obvious question is which of the countermeasures described above achieves the best tradeoff between efficiency (in terms of bandwidth, latency and server load) and effectiveness at disguising traffic.

Beyond the direct problem of disguising data, there are other obstacles to the widespread adoption of effective anonymization of Web traffic, including:

- How do users obtain and pay for the resources [19] (such as proxy server use) needed to anonymize their traffic?

- How do providers of anonymous services deal with their abuse [5] (for denial-of-service attacks, crime, or other undesired activity)? What is the right tradeoff between anonymity and accountability?
- How can anonymous services be adapted to real-life network complications such as firewalls and mobility?

References

[1] The Anonymizer. http://www.anonymizer.com.

[2] Google Fires New Salvo in Search Engine Size Wars. http://searchenginewatch.com/searchday/01/sd1211-google.html.

[3] Web Ad Blocking under Linux/Unix, BeOS, MacOS, and Windows. Available at http://www.ecst.csuchico.edu/~atman/spam/adblock.shtml.

[4] Zero-Knowledge Systems. http://www.zks.net.

[5] O. Berthold, H. Federrath, and S. Kopsell. Web MIXes: A System for Anonymous and Unobservable Internet Access. In H. Federrath, editor, *Designing Privacy Enhancing Technolgoies*, volume 2009 of *LNCS*, pages 115–129. Springer-Verlag, 2001.

[6] D. L. Chaum. Untraceable Electronic Mail, Return addresses, and Digital Pseudonyms. *CACM*, 1981.

[7] D. L. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *J. Cryptology*, 1:65–75, 1988.

[8] H. Cheng and R. Avnur. Traffic Analysis of SSL-encrypted Web Browsing. Available at: http://www.cs.berkeley.edu/~daw/teaching/cs261-f98/projects/final-reports/ronathan-heyning.ps.

[9] G. Danezis. Traffic Analysis of the TLS Protocol. Available at: http://www.cl.cam.ac.uk/~gd216/TLSanon.pdf.

[10] T. Dierks and C. Allen. The TLS Protocol Version 1.0. Internet RFC 2246, Janurary 1999.

[11] E. W. Felten and M. A. Schneider. Timing Attacks on Web Privacy. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, November 2000.

[12] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. Internet RFC 2616, June 1999.

[13] A. O. Freier, P. Karlton, and P. C. Kocher. The SSL Protocol Version 3.0. Internet Draft, March 1996.

[14] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Onion Routing for Anonymous and Private Internet Connections. *CACM*, 42(2), February 1999.

[15] E. Grabber, P. B. Gibbons, D. M. Kristol, Y. Matias, and A. Meyer. Consistent, Yet Anonymous, Web Access with LPWA. *CACM*, 42(2), February 1999.

[16] T. H. Haveliwala, A. Gionis, and P. Indyk. Scalable Techniques for Clustering the Web. In *WebDB (Informal Proceedings)*, pages 129–134, 2000. (http://dbpubs.stanford.edu:8090/pub/2000-23).

[17] K. Holtman and A. Mutz. Transparent Content Negotiation in HTTP. Internet RFC 2295, March 1998.

[18] T. Kan and S. Scherf. The audio CD database. http://www.cddb.com.

[19] C. Molina and L. F. Marshall. True Anonymity Without Mixes. In *Proc. IEEE Workshop on Internet Applications*, July 2001.

[20] K. Moore and N. Freed. Use of HTTP State Management. Internet RFC 2964, October 2000.

[21] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[22] V. N. Padmanabhan and L. Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. In *Proc. of ACM SIGCOMM*, San Diego, CA, USA, August 2001.

[23] C. W. Rackoff and D. R. Simon. Cryptographic Defense against Traffic Analysis. In *Proc. of* 25th *ACM Symposium on Theory of Computing*, May 1993.

[24] J. Raymond. Traffic Analysis: Protocols, Attacks, Design Issues and Open Problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 10–29. Springer-Verlag, 2001.

[25] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Secruity*, 1(1):66–92, November 1998.

[26] C. Shields and B. N. Levine. A Protocol for Anonymous Communications Over Internet. In *Proc. of* 7th *ACM Conference on Computer and Communication Security*, November 2000.

[27] P. F. Syverson, G. Tsudik, M. G. Reed, and C. E. Landwehr. Towards an Analysis of Onion Routing Security. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, volume 2009 of *LNCS*, pages 96–114. Springer-Verlag, 2001.

[28] C. J. van Rijsbergen. *Information Retrieval. 2nd ed.* Butterworths, 1979.