

Taking TrackBack Back (from Spam)

Paul Gerecht Rob McDonald Dan Sandler Andy Thomas-Stivalet
Dan S. Wallach

{gerecht, robmcd, dsandler, creepy, dwallach}@rice.edu

Rice University, 6100 Main Street, MS-132, Houston, TX 77005, USA.

Abstract

The TrackBack protocol, conceived as a way to automatically link together web sites which reference one another, has become a new vector for spammers wishing to divert web surfers to their sites. A site which supports TrackBack allows any entity to inject arbitrary HTML code, plus the URL of the sender, into its pages; an attacker need only follow the TrackBack protocol to exploit the system and leverage such a site in a link farm. Current approaches to combating TrackBack spam are limited to content-based filters (of the sort currently used against email and weblog comment spam). In this paper, we propose a way to identify TrackBack spam by considering the relationship between the sender's URL and the site under attack. In particular, we observe that, for spam TrackBacks, the page at the given URL does not link to the page to which the TrackBack was sent. We have developed software for weblog authors that rejects TrackBacks from sources lacking this reciprocal link. Data collected from our users demonstrates that this test is 100% accurate at identifying and separating spam from legitimate TrackBacks.

1 Introduction

Weblogs (or “blogs”) are widely used for purposes ranging from personal diaries to professional journalism. The blog search engine Technorati¹ claims to index over 35 million web sites [19] which fit the profile — pages containing short articles, periodically updated, and chronologically organized. Weblogs also document and annotate one another; most blogging software allows readers to append *comments* to the text of individual articles, turning each blog post into its own thread of conversation. Bloggers also tend to reply to others with entries on their own blogs, linking to the source material and offering additional thoughts. This pattern makes it easy to follow a conversation in reverse (following links to earlier articles), but does not easily facilitate reading the conversation chronologically.

¹<http://www.technorati.com/>

The TrackBack protocol [20] provides for a kind of automatic cross-linking in this scenario. Figure 1 illustrates the chain of events that cause a TrackBack to be created; when Alice writes about Bob's blog, her blogging software sends a TrackBack request to Bob so he can automatically create a reverse link to Alice. Now the conversation may be followed in either direction, increasing the dense interconnectedness (what Nelson called “intertwining” [15, 16]) of weblogs.

Figure 2 shows the dark side of TrackBack: it is trivial to abuse the protocol by posting unrelated (and unwanted) HTML and URLs to the millions of blogs which support TrackBack. In fact, it is estimated that 98% of all TrackBacks are spam [14]. Because the protocol is designed for unattended operation, it does not lend itself to CAPTCHA [23] or similar tests requiring human interaction. Legitimate users of TrackBack do not necessarily have any preexisting relationship, so traditional authentication techniques are similarly unhelpful. Current approaches to combating TrackBack spam are therefore limited to content-based filters akin to those used against email spam as well as spam in blog comments.

In this paper, we propose a way to identify TrackBack spam by considering the relationship between the sender's URL and the site under attack. In particular, we observe that, for spam TrackBacks, the page at the given URL does not link to the page to which the TrackBack was sent. By rejecting TrackBacks whose sources lack this reciprocal link, we can thus block TrackBack spams. We have developed software for weblog authors which performs this test and flags TrackBacks accordingly; this test requires no changes to the underlying TrackBack protocol and is therefore compatible with all existing legitimate sources of TrackBack. Data collected from our users indicates that our technique successfully blocks all current TrackBack spam with no errors of any kind.

The rest of this paper is organized as follows: we present background in Section 2; we detail the design of our solution in Section 3 and evaluate its performance in Section 4; Section 5 discusses our findings and Section 6 concludes.

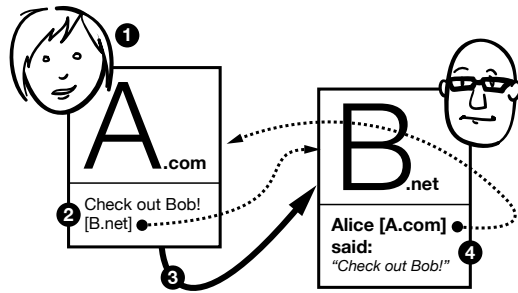


Figure 1. The TrackBack lifecycle. ❶ Alice sees an interesting article on Bob's weblog. ❷ Alice writes her own article, linking to B.net. ❸ Alice's weblog software sends a TrackBack message to Bob's weblog, containing Alice's name, the URL of her article, and a short comment. ❹ Bob's weblog software receives the TrackBack and inserts this information on B.net.

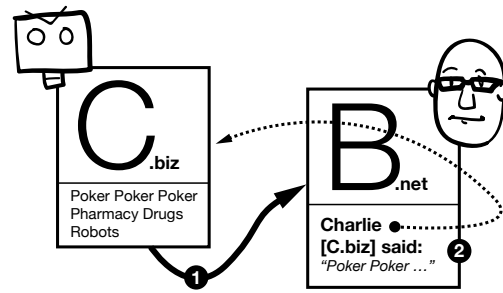


Figure 2. TrackBack spam. ❶ Charlie writes software to send irrelevant TrackBacks to many web sites, including Bob's. ❷ Bob's weblog software obligingly inserts Charlie's spam text, along with a link back to C.biz.

2 Background

2.1 TrackBack

The TrackBack protocol was developed in 2002 by SixApart for their Movable Type² weblog platform. It is now supported by most major stand-alone (i.e., not hosted, as with LiveJournal.com) weblog software, including WordPress³. A single TrackBack message (or “ping”) consists of an HTTP POST to the *ping URL* containing, at a minimum, a URL representing the source of the ping. Optionally, the ping may also contain three other strings: a title, the name of the source, and an HTML excerpt. The server receiving the ping responds with a small fragment of XML indicating the success of the operation; typically the server will act on the received ping by posting its contents to a web page (e.g., in the form of a blog comment).

2.2 TrackBack Spam

Internet link spammers, like email spammers, cast a very wide net; they look for any way to inject links into Web pages, be it through posting to guestbooks and forums, or even submitting any arbitrary HTML form they find (hoping the submission winds up on a web site somewhere)⁴. The spammer's goal is to get just a few click-throughs, or to bump a web site up just a few notches on search indexes. (In a system like Google's PageRank [3], the accumulated importance of every link's source will influence the importance of the link's destination, so a “search engine optimizer” wants to create as many

inbound links as possible, particularly from important web pages [24].) In current TrackBack implementations, none of the fields in a ping is validated for correctness, which means it is trivial to abuse TrackBacks to create “link farms” which can increase the target site's ranking.

To remove this particular incentive, several large search engines collaborated in 2005 to propose `rel="nofollow"`, an HTML extension to the `<a>` tag that instructs web search engines not to index the referenced page nor to assign it importance based on the source URL [10, 5]. The intent (as it pertains to TrackBack spam) is that weblog software will use `nofollow` when referencing any URL received in a TrackBack, and so spammers will be unable to use TrackBack-injected links to boost their search engine rankings. Unfortunately, this does not seem to have had the desired deterrent effect on spammers [1]; many blogs do not yet support `nofollow`, and those that do are still allowing spammers to use their pages as free advertising.

2.3 Spam Countermeasures

2.3.1 Turing tests, puzzles, challenges

It is instructive to consider the approaches that are effective for comment spam but are inapplicable to TrackBack. For example, automated “Turing tests” (e.g., CAPTCHAs [23]) force a user submitting a blog comment (or any Web form) to solve a “hard” AI problem, such as reading partially-corrupted characters from an image. This approach is infeasible for TrackBack, which by design functions without a human present.

Other promising ideas for filtering out spam would require changes to the underlying TrackBack protocol, making them incompatible with the millions of existing systems that support TrackBack. Email challenges,

²<http://www.movabletype.com/>

³<http://wordpress.org/>

⁴For example, we have observed spam submissions to our course registration web form, filling out every field with advertisement text.

in which users must supply a valid e-mail address (and reply to messages sent to that address) when posting comments, are quite successful at stopping automated form-submission robots and simple blog spammers. The TrackBack protocol does not currently allow an email address to be specified with each ping, and so any server wishing to require email challenges will necessarily be incompatible with the millions of TrackBack clients already in existence. Similarly, crypto puzzles like Hashcash [2] and other proofs of computational work [9] are impractical if backward-compatibility is desired.

2.3.2 Moderation

Many blog authors place all TrackBacks (and comments) into a moderation queue, to be filtered by hand; this labor-intensive approach does not scale as the volume of spam increases. Some blog authors have become sufficiently frustrated and have disabled TrackBack altogether [25]. Others have declared TrackBack to be “dead” (see, e.g., [6, 21]) because of the spam problem.⁵

2.3.3 Content-based filtering

It seems natural to treat TrackBack spam like email spam, which is typically identified by analyzing the message content for its “spamminess.” Early TrackBack spam filters used a naïve list of blacklisted words, but spammers quickly adopted the same tricks used by their email counterparts to defeat simple pattern matching filters.

More recently, Bayesian techniques have been borrowed from anti-spam solutions for email, but it is unclear whether these will work as well in the TrackBack realm. Graham [11] popularized the use of Bayesian inference models for spam classification (see also [18]), yet he notes that the power of this technique will force spammers to remove all observably “spammy” words from their emails. In particular:

[...] the spam of the future will probably look something like this:

```
Hey there. Thought
you should check
out the following:
http://www.27meg.com/foo
```

because that is about as much sales pitch as content-based filtering will leave the spammer room to make. [11]

Somewhat unsurprisingly, this closely matches the format of a modern TrackBack spam; for example, the following TrackBack comes from our data and is representative of current spams:

URL: [http://\[redacted\]/](http://[redacted]/)

Comment: I am so thrilled to find this being served for FREE. This is the greatest discovery I have ever made on the web.

Spammers have even begun to use text from the victim’s weblog itself as the content of the message, thus increasing the likelihood of using tokens that are relevant (and therefore highly associated with non-spam) [8]. This ability is not available to email spammers, who can’t know *a priori* what tokens will be most relevant to any given victim. Content-based classification is therefore likely to be less effective on TrackBack spam than it is on email spam.

The current best practice for the hapless weblog author is to install one of a handful of “kitchen sink” anti-spam plug-ins for his blogging package. These tools employ a combination of the above approaches to attempt to automatically flag spam. For example, Spam Karma⁶ uses blacklisted words, IP addresses, CAPTCHA, and other local tests on weblog comments and TrackBacks. Akismet,⁷ by contrast, is a centralized service which operates as a black-box predicate (legit or spam?) for any submitted comment or TrackBack. The tests Akismet runs are not made public; we cannot determine if they use content filtering or even if they use an approach like the one we propose here. (We can determine, however, that Akismet presents potential trust and availability problems; the server could cease to function, or it could decide unilaterally and without warning on a different definition of “spam.” It is vastly preferable for individual TrackBack receivers to be able to classify spam without external assistance.)

2.4 Pingback

No discussion of TrackBack is complete without mentioning Pingback [13]. Pingback serves nearly the same purpose as TrackBack, with different details: the ping’s POST payload is formatted as XML-RPC instead of CGI arguments; a Pingback ping contains only the URL of the sender’s article, and no other information. This means Pingback is slightly less flexible; the sender of a Pingback may not specify an excerpt or comment in the ping. In fact, a Pingback recipient must, in order to display anything meaningful on a web site (for example, some kind of excerpt of the source page), fetch the resource at the given URL and analyze its contents. The 1.0 specification suggests, but does not require, that Pingback recipients check that page for an inbound link in order to curb abuse. In the following sections we will apply this very test to TrackBack, and show how effective it is at eliminating spam.

⁵We believe these reports to be greatly exaggerated, hence this paper.

⁶<http://unknowngenius.com/blog/wordpress/spam-karma/>

⁷<http://www.akismet.com/>

3 TrackBack Validation

Rather than analyzing the textual content of the TrackBack (which may be misleading or absent), or adding requirements to the TrackBack messaging protocol (which would deny service to existing blogs speaking the old protocol), we instead examine the only required part of the TrackBack: its URL payload. Every spam TrackBack we have seen includes a URL pointing to an HTML page that contains no reciprocal links back to the TrackBack recipient. In the example shown by Figure 2, Charlie doesn't bother to update his advertising web site to link to any of his (potentially thousands of) victims. On the other hand, legitimate TrackBacks between weblogs, such as those illustrated in Figure 1, are accompanied by a stable link to the TrackBack recipient. Although the TrackBack specification does not currently require this reciprocal link, common practice among webloggers is always to provide it.

Therefore, a TrackBack recipient can identify spam by the lack of an inbound link from the supplied URL. To test this hypothesis, we developed a software plug-in for WordPress weblogs called the TrackBack Validator.⁸ The Validator modifies the blog's behavior to only admit TrackBacks whose URL includes a reciprocal link. The revised TrackBack lifecycle is illustrated by Figure 3.

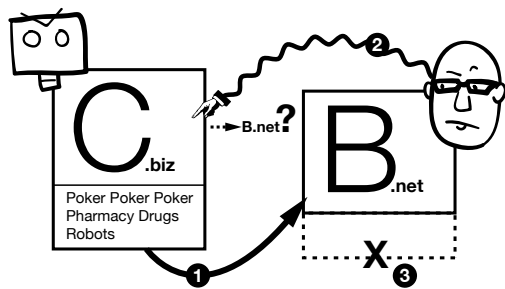


Figure 3. TrackBack validation. ❶ As before, Charlie sends a spam TrackBack to Bob's weblog. ❷ Before placing the contents of the TrackBack on his page, Bob's weblog software first examines C.com (the URL contained in the TrackBack) for links to B.net. ❸ Failing to find any such links, Bob decides to treat the TrackBack as spam, and does not post its contents to B.net.

Our plug-in also allows the blog owner to periodically re-validate old (previously validated) TrackBacks, to prevent spammers from setting up short-lived link pages to thwart this validation step. (We examine in more detail what happens when spammers start to set up longer-lived link pages in Section 5.)

Bloggers who use the Validator plug-in have the option to submit data about their site's TrackBack spam to us for research purposes. As of this writing, 69 such sites

⁸Our plug-in can be downloaded from <http://seclab.cs.rice.edu/proj/trackback>.

have contributed almost 45,000 spam TrackBacks to our database. Our analysis of this data follows.

4 Evaluation

4.1 Accuracy and Deterrent Effects

The accuracy of the TrackBack Validation scheme described in Figure 3 has proven to be remarkably successful; out of the 45,000 TrackBack requests that it identified as spam, there was not a single legitimate false positive, nor a report from our users of a false negative. Every piece of empirical data collected suggests that the TrackBack Validation method does not mistakenly classify legitimate TrackBack requests as spam.

Our only known false positive of any kind related to a bug in an early version of our plug-in. In anticipating possible counterattacks (of the sort we discuss in Section 5), we flagged as spam any TrackBack whose URL contained the recipient's URL anywhere inside. (Such a scheme might give a spammer a lightweight way to create a page which dynamically links to the given URL in order to fool our Validator. For example, the URL <http://spam.com/ad?from=http://legit.org> contains enough information for a stateless CGI script to construct a link to legit.org.) One of our users quickly discovered that our algorithm also blocked legitimate TrackBacks sent from a site to itself, because the site's own URL is obviously a substring of any such TrackBacks. We fixed the plug-in, and have heard of no further false positives.

The study of the habits of these spammers has shown that some take an active interest in the success or failure of their TrackBack requests. In Figure 4, the spam TrackBack requests received by the four most heavily-spammed weblogs in this experiment are shown by week where, for each weblog, the start of the first week is the day upon which they initially install the TrackBack Validator plug-in. The trends in Figure 4 show that the amount of spam sent by these entities decreases dramatically over time as they realize their attempts to get their spam links posted via TrackBack are ineffective. However, not all of the spammers exhibit this behavior; their solicitations continued regardless of the successes or failures of their previous spam TrackBack requests.

4.2 Observations

While it was not our goal to characterize the nature of TrackBack spam, we were able to gather some interesting data during the course of verifying the accuracy of the Validator plug-in.

By using tools such as traceroute, HostIP⁹, and DNSStuff¹⁰, it was possible to determine the location of

⁹<http://api.hostip.info>

¹⁰<http://www.dnsstuff.com>

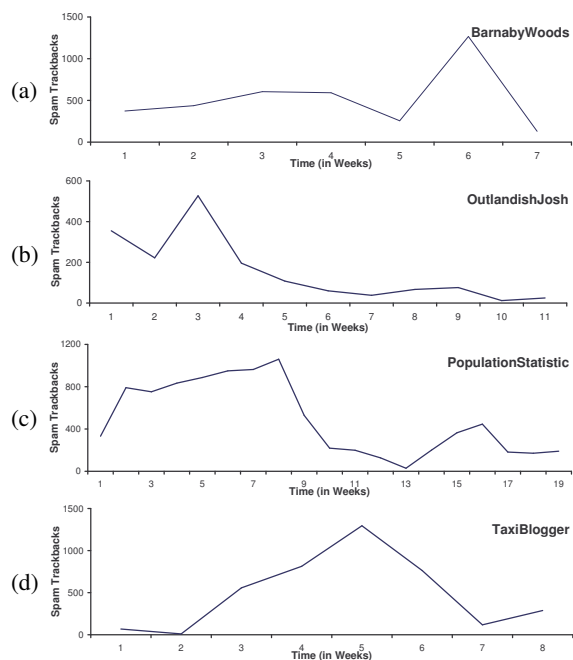


Figure 4. TrackBack spam by week, for several weblogs. Each weblog has reported data over a different time period.

the various hosts sending spam TrackBack requests to all the weblogs participating in this experiment with a fairly high degree of certainty. These results are shown in Figure 5.

| Country | % | Country | % |
|--------------------|-------|----------------|-------|
| Russian Federation | 19.19 | Brazil | 2.01 |
| United States | 16.88 | Germany | 1.76 |
| Korea | 7.49 | United Kingdom | 1.62 |
| Ukraine | 6.67 | Taiwan | 1.43 |
| Japan | 5.54 | Ireland | 1.40 |
| European Union | 3.77 | India | 1.33 |
| Malaysia | 3.27 | Nicaragua | 1.17 |
| China | 2.55 | Thailand | 1.11 |
| Costa Rica | 2.25 | France | 1.10 |
| Vietnam | 2.04 | Mexico | 1.09 |
| | | (104 others) | 16.32 |

Figure 5. TrackBack spam by originating country. Each country’s contribution, as percent of total TrackBack spam (rounded to the nearest hundredth of a percent), is shown for all sites in this study. Countries contributing 1.0 percent or less not shown.

We also considered the textual content of TrackBack spam. Many independent victims often received very similar TrackBacks spams, including similar or identical text and URLs (including typographical errors and other “chaff” designed to thwart content-filters), from disparate IP addresses. This leads us to believe that, just as email spammers do, TrackBack spammers rely on botnets: innocent PCs around the Internet under control of

the spammer due to virus infection or other remote security exploit. Attempts to estimate the number of active TrackBack spammers based on these recurring spam profiles are beyond the scope of this paper.

5 Discussion

Our TrackBack Validation plug-in has been shown to work effectively in combating TrackBack spam. Spammers who wish to overcome our mechanism are forced to indefinitely maintain reciprocal links from their own web sites, effectively increasing their necessary investment of time and resources. Furthermore, the spammer’s site, by linking to its victims, will actually benefit the victims’ search engine rankings by sharing part of the spammer’s ranking with each of its victims. Best of all, if the spammer is effectively publishing a list of its victims, that list would provide compelling evidence that could be used against the spammer in legal proceedings.

In the limit, we are effectively pushing spammers to run “legitimate” weblogs. If spammers’ weblogs are following the TrackBack protocol correctly and are legitimately providing reciprocal links, then we face a more fundamental question: is such a TrackBack message actually spam? If a “real” blog is linking to the victim, regardless of any spam-like content it might contain, then the TrackBack the victim receives could well be defined as “legitimate.” At that point, the issue is not one of spam vs. non-spam, but rather one of *relevance*. If a blog is receiving legitimate TrackBacks from irrelevant blogs, then it would certainly be desirable to do some kind filtering. There are two obvious approaches to pursue:

- The blog’s own readers can be used to weed out irrelevant posts. This approach has been used very effectively on Slashdot.org, among other popular web sites. While such group moderation approaches will not eliminate the spam entirely, they will ensure that few users ever see it.
- While the contents of a particular TrackBack spam (or comment spam) may be insufficient to classify it with statistical techniques, the TrackBack’s hyperlink could be followed and the contents on the remote web server could be used for statistical classification of the TrackBack. Graham [12] argues that this approach will not only improve classification, but will also *punish* the spammer with the sheer volume of traffic from each of the victims scanning the spammer’s web site.

Furthermore, by forcing spammers to have “legitimate” web sites, the spammers lose some benefit they presently gain from using botnets. In particular, simple blacklisting might be effective against spammers with stable web sites, whereas blacklisting is ineffective against bots, since the same bot may never be seen twice.

We note that the spammer's web site might generate customized pages (as observed by Burton [4]) as a function of which weblog site is querying its pages, in a fashion similar to search engine optimizers, who might feed different content to search engines than to normal users [24]. Google addresses this issue by having a *death penalty*—if a site is ever caught behaving in this fashion, it will be removed from the search engine. Similarly, weblogs might need to distinguish whether they were reading customized content. An anonymous communication system like Tor [7] could be used to hide the IP address of the victim as it probes the spammer's URL for reciprocal links, although the URL would need to be stripped of any identifying information.

6 Conclusion

We conclude, both from the anecdotal experiences of our users as well as the empirical data from their blogs, that link validation effectively eliminates all current TrackBack spam. For a system like TrackBack, which is demonstrably useful in interconnecting weblogs but suffers from a signal-to-noise ratio of 1:50, this is a tremendously important development and may very well serve to save TrackBack.

Recently, the authors of TrackBack have formed the TrackBack Working Group [22] to formulate and submit a revised version of TrackBack as an IETF standard. One of the stated goals of the standardization process is to address problems in the existing system, including TrackBack spam [17]. We are pleased that the working group is currently considering the technique we describe for inclusion in the new standard.

Acknowledgments

We would like to acknowledge the thoughtful comments we received from members of the WordPress-Hackers and TrackBack Working Group mailing lists, and we are especially indebted to the many bloggers who provided spam data for our research.

This work was supported, in part, by NSF grants CNS-0524211 and CNS-0509297 as well as generous gifts from IBM, Microsoft, and Schlumberger.

References

- [1] C. Arthur. Interview with a link spammer. *The Register*, Jan. 2005. http://www.theregister.co.uk/2005/01/31/link_spamer_interview/.
- [2] A. Back. Hashcash - A denial of service counter-measure, Aug. 2002. <http://hashcash.org/papers/hashcash.pdf>.
- [3] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107-117, 1998.
- [4] K. Burton. Nasty new trackback spam. *Kevin Burton's Feed Blog*, Nov. 2005. http://feedblog.org/2005/11/nasty_new_track.html.
- [5] T. Çelik, K. Marks, M. Cutts, and J. Shellen. `rel="nofollow"` draft specification, Jan. 2005. <http://microformats.org/wiki/relnofollow>.
- [6] T. Coates. Trackback is dead. Are comments dead too? *plasticbag.org*, Apr. 2005. http://www.plasticbag.org/archives/2005/04/trackback_is_dead_are_comments_dead_too.shtml.
- [7] R. Dingleline, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, Aug. 2004.
- [8] DrDave. The State of Spam [Karma], Jan. 2006. <http://unknowngenius.com/blog/archives/2006/01/30/the-state-of-spam-karma/>.
- [9] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In *Proc. CRYPTO 1992, Lecture Notes in Computer Science*, volume 740, pages 139-147. Springer, 1992.
- [10] Google Blog. Preventing comment spam. *Official Google Blog*, Jan. 2005. <http://googleblog.blogspot.com/2005/01/preventing-comment-spam.html>.
- [11] P. Graham. A plan for spam, Aug. 2002. <http://paulgraham.com/spam.html>.
- [12] P. Graham. Filters that fight back, Aug. 2003. <http://paulgraham.com/ffb.html>.
- [13] S. Langridge and I. Hickson. Pingback 1.0, 2002. <http://hixie.ch/specs/pingback/pingback>.
- [14] M. Mullenweg. Trackback spam stats. By email to TrackBack working group list, Feb. 2006. <http://www.sixapart.com/pipermail/trackback-protocol/2006-February/000088.html>.
- [15] T. H. Nelson. *Computer Lib/Dream Machines*. Microsoft Press, Redmond, WA, USA, 1987.
- [16] E. S. Raymond. Intertwingled. *The Jargon File, version 4.4.7*, Dec. 2003. <http://www.catb.org/jargon/html/i/intertwingled.html>.
- [17] B. Reese. Submitting trackback as an internet standard. *Six Apart ProNet*, Feb. 2006. http://www.sixapart.com/pronet/weblog/2006/02/submitting_trac.html.
- [18] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A Bayesian approach to filtering junk e-mail. In *AAAI-98 Workshop on Learning for Text Categorization*, pages 55-62, 1998.
- [19] D. Sifry. State of the blogosphere, april 2006 part 1: On blogosphere growth. *Sifry's Alerts*, Apr. 2006. <http://www.sifry.com/alerts/archives/000432.html>.
- [20] SixApart. TrackBack technical specification, 2002-2004. http://www.sixapart.com/pronet/docs/trackback_spec.
- [21] R. Tanglao. Trackback really is dead - sorry Elliott et al, May 2005. http://www.rolandtanglao.com/archives/2005/05/01/trackback_really_is_dead_sorry_elliott_et_al.
- [22] TrackBack working group. <http://www.lifewiki.net/trackback/>.
- [23] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of EURO-CRYPT*, pages 294-311, Warsaw, Poland, May 2003.
- [24] Wikipedia. Search engine optimization (*version dated 18 apr 2006*). http://en.wikipedia.org/wiki/Search_engine_optimization.
- [25] J. Zawodny. Trackback is dead, Aug. 2005. <http://jeremy.zawodny.com/blog/archives/005049.html>.