(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

PCT

(43) International Publication Date
12 July 2007 (12.07.2007)

(10) International Publication Number
WO 2007/076715 A1

(51) **International Patent Classification:**
G06F 11/28 (2006.01) *H04L 9/00* (2006.01)
G06F 15/16 (2006.01)

(21) **International Application Number:**
PCT/CN2006/003728

(22) **International Filing Date:**
30 December 2006 (30.12.2006)

(25) **Filing Language:** English

(26) **Publication Language:** English

(30) **Priority Data:**

60/766,115	30 December 2005 (30.12.2005)	US
60/766,119	31 December 2005 (31.12.2005)	US
11/616,950	28 December 2006 (28.12.2006)	US

(71) **Applicant** *(for all designated States except US):* **METASWARM (HONGKONG) LTD.** [CN/CN]; Unit C, 11/F, Gaylord Comm. Bldg., 114-118 Lockhart Road, Wanchai, Hong Kong (CN).

(72) **Inventors: SHANNON, Marvin;** 3579 East Foothill Blvd., #328, Pasadena, CA 91107 (US). **BOUDEVILLE, Wesley;** 33 Richardson Arcade, Winthrop, Perth 6155 (AU).

(74) **Agent: LIU, SHEN & ASSOCIATES;** A0601, Huibin Building, No.8 Beichen Dong Street, Chaoyang District, Beijing 100101 (CN).

(81) **Designated States** *(unless otherwise indicated, for every kind of national protection available):* AE, AG, AL, AM, AT, **AU, AZ,** BA, BB, **BG, BR, BW, BY, BZ,** CA, **CH,** CN, **CO, CR, CU, CZ,** DE, **DK, DM, DZ,** EC, EE, **EG,** ES, **FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ,** NA, **NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU,** SC, SD, SE, **SG, SK, SL, SM, SV, SY, TJ, TM,** TN, **TR,** TT, **TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW**

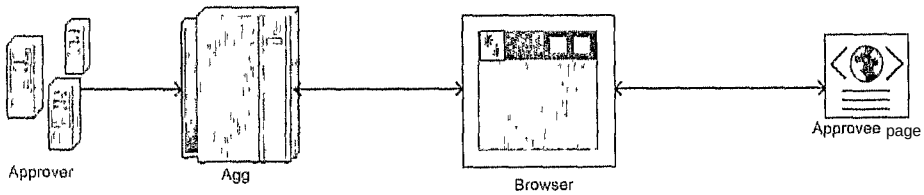
(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available):* ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, **DK,** EE, ES, FI, FR, GB, GR, HU, IE, **IS, IT,** LT, LU, LV, MC, NL, PL, PT, **RO,** SE, **SI,** SK, TR), OAPI (BF, **BJ,** CF, CG, CI, CM, GA, GN, GQ, GW, ML, **MR,** NE, SN, TD, TG).

Declaration under Rule 4.17:
— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(Hi))*

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEM AND METHOD OF APPROVING WEB PAGES AND ELECTRONIC MESSAGES



(57) Abstract: We expand our antiphishing methods to show how web pages and electronic messages might have multiple approval tags. These designate that the item was approved by other parties. Typically, the latter are customers of an Aggregator. When the item is viewed by a browser, the browser can have a plug-in parse the tags, and verify these by contacting the Aggregator. Useful for ascertaining that charitable websites and messages purporting to be from charities are real (or not). More generally, it can also be used to let web pages and messages have endorsements from other organizations. We show how a blog or journal that lets authors write articles in an unmoderated manner can let them write verifiable articles. A tag is added to an article, that points to another website, where a "Checker" runs. The tag also has an author id. A visitor to the blog has code that hashes an article and goes to the Checker and presents the (id, hash) and asks if this is valid. Previously, the author registered the hash with the Checker, as one of her valid hashes. This can be extended to binding the article to the blog page, by the author hashing her article and the page address, and registering this with the Checker. This prevents someone copying her article to another website and having it be verifiable there.

WO 2007/076715 A1

System and Method of Approving Web Pages and Electronic Messages

CROSS-REFERENCES TO RELATED APPLICATIONS

This application claims the benefit of the filing date of U.S. Provisional Application, Number 60/766119, "System and Method of Approving Web Pages and Electronic Messages", filed December 31, 2005, and which is incorporated by reference in its entirety. It also incorporates by reference in its entirety the U.S. Provisional Application, Number 60/766115, "System and Method of Verifying Blogging/Journaling", filed on December 30, 2005.

TECHNICAL FIELD

This invention relates generally to information delivery and management in a computer network. More particularly, the invention relates to techniques for letting people or organisations approve web pages, in a verifiable manner.

BACKGROUND OF THE INVENTION

As usage of the Internet continues to increase, so does the incidence of fraud on the Internet. A fraudster might make a website that claims to be accepting donations for a charitable cause. As a prominent example, after Hurricane Katrina, there were a slew of such websites. On 9 September 2005, CNN reported that 2300 websites about Katrina had been tallied by the FBI, who considered most of these to be bogus

(http://money.cnn.com/2005/09/09/pf/beware_disaster_scams/?cnn=y es).

Prior to Katrina, the Asian Tsunami of December 2004 also gave rise to fraudulent websites. It can be anticipated that future disasters will also lead to future fraudulent websites.

Nor are the frauds necessarily confined to websites. They might also involve the sending of mass electronic messages (spam). Purporting to be from a relief organization or even a government. If the messages are email, then it is trivial to forge the sender line to be whatever the author wishes. In general, the message body can have text claiming to be from that organization. But, typically, the message will also have a link which goes to a pharm (fake website).

The websites and messages involve social engineering. Through text and images (and possibly even audio and video), they try to fool the unwary reader. Especially possible if the web pages and messages are very professionally done. So that a casual reader cannot easily distinguish between a fake and a real charity. It may actually be impossible to objectively do so. (At least before this Invention.) That is, there is simply not enough information in the page or message for a manual perusal to objectively reach a conclusion.

This is reflected in advice currently given by antifraud groups. For instance, scambusters.org suggests 4 measures(http://www.scambusters.org/hurricanekatrinascams.html). The using of common sense. Never responding to an email request for a donation. Not opening attachments in those emails (because they might have viruses). And manually checking the charity's name against a list of real charities. Where the latter might be at give.org, which is run by the Better Business Bureau.

The US Government also offers similar advice. See for example http://www.fbi.gov/katrina.htm.

Another existing method involves companies like Verisign Corp. and Truste Coip. that offer seals that can be added to the web page of an organization approved by them. These seals might be clickable, and go back to those companies. So that a user could click on these and get some validation of the organization. But if the seals are not clickable, then a fraudster can just copy the images and put them into her pages or messages. If the seals are clickable, she might still do this and make the images non-clickable. Because not everyone who sees the seals knows that they should be clickable. Or, she might make them clickable, but the click goes back to her website, or another website run by her, and not the websites of the antifraud companies. Not all users are aware that clickable seals must go back to the latter. And even someone who does know, often does not bother to click them.

In a manner related to the above fraud problems, there has also been a long standing problem regarding authorship of articles in newsgroups and blogs. These are often not primarily financial fraud related, but may be regarded as an issue of fraudulant authorship. It arose with the first killer application of the Internet, email. This led to the proliferation of innumerable newsgroups by 1990. Often, anybody could post to such a group. Groups could be moderated or unmoderated.

Several years after the advent of the Web, blogging become popular. Often, a blog is associated with a given person, who writes the main articles in it. But usually, a blog invites comments by others, who might assume false identities. Newgroups, in the pre-Web sense, still exist. Now some offer the ability to also upload images, audio or video. Blogs can do this too.

Both often have a common problem. Anyone might make a submission. Perhaps furnishing a misleading address (like an email address or link). It would be very useful for a person viewing a blog or newsgroup (by using a browser, say) to somehow be able to see which entries are validated, in some sense.

A second problem also arises. Blogs and newsgroups, have been targeted by "bots". These are programs (i.e. robots) that write spam messages. Usually, a bot message advertises some good or service, with a link to the spammer's website. Often, the blog or newsgroup administrator and readers consider such submissions to be highly undesirable. This bot problem only appears if the blog is unmoderated. However, going to a moderated blog has drawbacks of its own. Namely the manual effort by a human moderator. Plus the time delay before a submission by a human gets posted to the blog

REFERENCES CITED

"We Blog: Publishing Online with Blogs" by P Bausch et al, Wiley 2002.

"Survey of Text Mining: Clustering, Classification and Retrieval" by M Berry, Springer 2003.

"Understanding Search Engines" by M Berry, SIAM 2005.

"The Weblog Handbook" by R Blood, Perseus 2002.

"The NAT Handbook" by B Dutcher, Wiley 200 1.

"Developing Feeds With RSS and Atom" by B Hammersley, O'Reilly 2005.

"XML in a Nutshell" by E Harold and W Means, O'Reilly 2004.

"Clustering for Data Mining" by B Mirkin, Chapman and Hall 2005.

"Search Engine Marketing" by M Moran and B Hunt, IBM Press 2005.

"Hacking RSS and Atom" by L Orchard, Wiley 2005.

"Introduction to Data Mining" by P Tan, Addison-Wesley 2005.

"Blog Marketing" by J Wright, McGraw-Hill 2005.

fbi.gov/katrina.htm

give.org

scambusters.org

truste.org

verisign.com

SUMMARY OF THE INVENTION

The foregoing has outlined some of the more pertinent objects and features of the present invention. These objects and features should be construed to be merely illustrative of some of the more prominent features and applications of the invention. Other beneficial results can be achieved by using the disclosed invention in a different manner or changing the invention as will be described. Thus, other objects and a fuller understanding of the invention may be had by referring to the following detailed description of the Preferred Embodiment.

We expand our antiphishing methods to show how web pages and electronic messages might have multiple approval tags. These designate that the item was approved by other parties. Typically, the latter are customers of an Aggregator. When the item is viewed by a browser, the browser can have a plug-in parse the tags, and verify these by contacting the Aggregator. Useful for ascertaining that charitable websites and messages purporting to be from charities are real (or not). More generally, it can also be used to let web pages and messages have endorsements from other organizations.

We show how a blog or journal that lets authors write articles in an unmoderated manner can let them write verifiable articles. A tag is added to an article, that points to another website, where a "Checker" runs. The tag also has an author id. A visitor to the blog has code that hashes an article and goes to the Checker and presents the (id, hash) and asks if this is valid. Previously, the author registered the hash with the Checker, as one of her valid hashes. This can be extended to binding the article to the blog page, by the author hashing her article and the page address, and registering this with the Checker. This prevents someone copying her article to another website and having it be verifiable there.

BRIEF DESCRIPTION OF THE DRAWINGS

There is one drawing. Figure 1 shows an Approver, an Aggregator, an approvee, and a browser looking at a page at the approvee. (The terms are defined in the text.)

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

What we claim as new and desire to secure by letters patent is set forth in the following claims.

We described a lightweight means of detecting phishing in electronic messages, or detecting fraudulent web sites in these earlier U.S. Provisional : Number 60522245 ("2245"), "System and Method to Detect Phishing and Verify Electronic Advertising", filed September 7, 2004; Number 60522458 ("2458"), "System and Method for Enhanced Detection of Phishing", filed October 4, 2004; Number 60552528 ("2528"), "System and Method for Finding Message Bodies in Web-Displayed Messaging", filed October 11, 2004; Number 60552640 ("2640"), "System and Method for Investigating Phishing Websites", filed October 22, 2004; Number 60552644 ("2644"), "System and Method for Detecting Phishing Messages in Sparse Data Communications", filed October 24, 2004; Number 60593114, "System and Method of Blocking Pornographic Websites and Content", filed December 12, 2004; Number 60593115, "System and Method for Attacking Malware in Electronic Messages", filed December 12, 2004; Number 60593 186, "System and Method for Making a Validated Search Engine", filed December 18, 2004; Number 60/593877 ("3877"), "System and Method for Improving Multiple Two Factor Usage", filed February 21, 2005; Number 60/593878 ("3878"), "System and Method for Registered and Authenticated Electronic Messages", filed February 21, 2005; Number 60/593879 ("3879"), "System and Method of Mobile Anti-Pharming", filed February 21, 2005; Number 60/594043 ("4043"), "System

and Method for Upgrading an Anonymizer for Mobile Anti-Pharming", filed March 7, 2005; Number 60/594051 ("4051"), "System and Method for Using a Browser Plug-in to Combat Click Fraud", filed March 7, 2005.

We will refer to these collectively as the "Antiphishing Provisionals".

Below, we will refer to the following U.S. Provisionals submitted by us, where these concern primarily antispam methods: Number 60320046 ("0046"), "System and Method for the Classification of Electronic Communications", filed March 24, 2003; Number 60481745 ("1745"), "System and Method for the Algorithmic Categorization and Grouping of Electronic Communications, filed December 5, 2003; Number 60481789, "System and Method for the Algorithmic Disposition of Electronic Communications", filed December 14, 2003; Number 60481899, "Systems and Method for Advanced Statistical Categorization of Electronic Communications", filed January 15, 2004; Number 60521014 ("1014"), "Systems and Method for the Correlations of Electronic Communications", filed February 5, 2004; Number 60521 174 ("11 74"), "System and Method for Finding and Using Styles in Electronic Communications", filed March 3, 2004; Number 60521622 ("1622"), "System and Method for Using a Domain Cloaking to Correlate the Various Domains Related to Electronic Messages", filed June 7, 2004; Number 60521698 ("1698"), "System and Method Relating to Dynamically Constructed Addresses in Electronic Messages", filed June 20, 2004; Number 60521942 ("1942"), "System and Method to Categorize Electronic Messages by Graphical Analysis", filed July 23, 2004; Number 605221 13 ("21 13"), "System and Method to Detect Spammer Probe Accounts", filed August 17, 2004; Number 60522244 ("2244"), "System and Method to Rank Electronic Messages", filed September 7, 2004.

We will refer to these collectively as the "Antispam Provisionals".

1. Approving Pages and Messages

Note that the above background advice involves entirely manual steps by a person using a browser or reading a message. These have drawbacks. Notably, she might not be aware of these steps. And even if she has some knowledge of these, finding a reputable website with a list of valid charities is not obvious. The give.org website mentioned above is not a prominent website, for example.

In this Invention, we describe an automated way to detect in an objective and lightweight fashion valid charitable websites and electronic messages. For the latter, we will choose the common example of email. Though our method can be applied to other types of electronic messaging, like faxes, Instant Messaging and SMS.

We describe the use of a browser and a custom plug-in that implements our method on the client desktop. Alternatively, the functionality of the plug-in might be incorporated into a browser. Also, our method is applicable in other client programs that can display hypertext documents and electronic messages.

In our Antiphishing Provisionals, we described an antiphishing method which involved the use of an Aggregator (or set of these), a browser plug-in, Partner Lists and a Notphish tag. The latter would be used in an email or web page, written perhaps as

<notphish a="bank0.com" />

Here, the email or page claims to be from BankO.

The plug-in would detect this tag, and extract domains from any links in the page or message, and possibly find a hash of that page or message. It would contact an Aggregator. To see if BankO was one of the Aggregator's customers. If so, then it would compare the domains with those of BankO's Partner List. Or, instead of or perhaps in addition, it would see if the hash was in a list of valid hashes for BankO.

Here, we extend the above idea. Suppose a web page is for a charity, with a domain charityO.org. Or perhaps there is a message from this charity. In either case, it can embed a new "Approval" tag. The representation depends on an actual implementation, and here we use one such representation, in these examples:

<approve from="agency.gov" to="charityO.org" />
<approve from="charity5.org" to="charityO.org" />

The "from" attribute designates another organization that approves of CharityO. The "to" attribute designates the organization being approved; in this case, CharityO. The Approval tag might have other fields, as described later.

The page or message could have several Approval tags. A key difference from the Notphish tag. The latter inherently could or should appear only once in a page or message. Because the Notphish tag is used to identify the owner of the page or message. There is only one such owner. Whereas Approval can be done by, several other parties.

Note that when a plug-in/browser is looking at a message, it is at a web page of some message provider that has received the message for the user. We distinguish between this and when the browser is looking at a generic non-message provider web page, by using the method of "2528".

The organizations mentioned in the from field are assumed to be customers of the Aggregator. Similar to our earlier methods. Because the Aggregator acts to accept only reputable organizations, that it has validated by some means. This role is vital. If the plug-in were to go directly to the addresses of the organizations mentioned in the from field, then a phisher ("Amy") could merely make a website that approves her main website that solicits donations.

(An alternative implementation is for the plug-in to have a hardwired list of reputable approval organizations. And it goes to those directly, without querying an Aggregator. Another alternative implementation is for the plug-in to periodically get such a list from an Aggregator, and then use it to directly query the organizations in it.)

If the plug-in finds a hash of the page or message, it should preferably first remove all the Approval tags. This lets the page or message be approved by several organizations. Since adding these tags will not change the hash.

A difference from the earlier Provisionals is that here, CharityO need not be a customer of the Aggregator. Though the organizations that approve it need to be customers, and hence validated, there is no such requirement for CharityO. This lets new organizations use our method for approval. It also lets each approver decide, using its own methods, which organizations it will approve.

An approver sends to the Aggregator a list of organizations that it approves ("approvees"). Typically, this list might have the base domains of the approvees. Optionally, the approver might furnish more detail about an approvee. It might say that a page or message from the latter can only have links with base domains in a list specified by the approver. Analogous to the Partner List of "2245". The approver might also give a list of hashes of pages or messages that it approves of.

Given the above information, the plug-in can then apply it to the page or message. However, an important difference arises, depending on whether the browser is looking at a page or message. Suppose the browser is at an URL in the charityO.org domain. Then if an approver just gave the base domain, charityO.org, the plug-in can indicate that the page is approved by the approver. Because the plug-in can take the URL and extract its base domain and compare this with the base domains approved by the approver.

Now imagine we are looking at an email with a sender address at charityO.org. If the approver only approved the base domain, charityO.org, then preferably, the plug-in should not say that the message was approved. Because the sender field can be trivially altered. An alternative is for the plug-in to indicate approval, but only if there are no links, or, if there are links, that these all have the base domain charityO.org. We recommend that this alternative be deprecated. Because a phisher might write a message with no links, where the text indicates that the reader should perhaps manually call a phone number, or type in an URL not at charityO.org.

In general, for approving messages, the preferred implementation is for the approver to have a list of valid domains in links in those messages or hashes.

The approver could also attach a comment to an approvee, or to a given page or message by the latter. The comment would be sent to the Aggregator, and then downloaded by the plug-in. Hence, the plug-in could make these available to the user, if she asked for them, for example.

Plus, the approver might also attach a number or string to an approvee, or to a given page or message by the latter. Without loss of generality, assume it is a number. The possible values this might take, and the meanings of those values, could be promulgated by the Aggregator. Hence, the plug-in could take these values, and summarize or display them (or their meanings) in their entirety, if the user asked for these. In terms of a summary, if it is meaningful to take a numerical average, then this might be done, for example.

Possibly, an approver might furnish several numbers or strings, for a given approvee or given page or message. Each might be for a different property or metric. For example, an approver might have a string that indicates a country code, or several country codes. The meaning of these could be that the approver is saying, or suggesting, that the approvee can only raise funds in those countries, or

from citizens or residents of those countries. The approver might be a government body with the authority to regulate charities, for instance.

Another example is a number that indicates how "strongly" the approver vouches for the approvee. One value is that the approvee is the approver itself. Another value could be that the approver has inspected the approvee's books. Another value might be that the approver only knows casually of the approvee.

Another example is a number that measures the urgency of the appeal. For example, an approvee raising money for cancer research might have moderate urgency. Since it makes little difference whether a donation is made today or two months from now, given the long time scales (years) for a suitable drug or therapy to be found and tested. Whereas an approvee raising money for famine victims might have greater urgency.

Another example is a number that measures the efficiency of the approvee in using donated funds. That is, the lower its overhead, the greater its efficiency. Bona fide charities might have to, or might voluntarily, reveal their overheads to an approver. Especially if the latter is a government regulator.

Another example is an expiration date for the approval. An approver might be reluctant to give an open-ended approval. So rather than having to decide at some future time whether to withdraw the approval, it simply defines an expiration. Which can tllen be implemented by the Aggregator.

When the plug-in analyzes the Approval tags, suppose it finds that a tag is wrong, but that several tags check out correctly? The plug-in might have a policy that if any tag is wrong, then it will take steps to alert the user, like coloring an icon in the toolbar, as was done in our antiphishing inventions. It could also turn off any links in the message. Including any "submit" buttons. The latter is important, to prevent the user filling in a form with her personal information and

pressing that button, which might then send it to the fraudster.

The plug-in might have a more permissive policy, in which it will do the above only if 2 tags are wrong, or if a majority of tags are wrong.

Now consider more closely what it means if the plug-in were to find out from the Aggregator that a tag is wrong. There could be degrees of wrongness, and a code could be returned to the plug-in that indicates why the tag is wrong. Firstly, the approver might not be a customer of the Aggregator. Which strongly suggests that the tag is inherently false. Likewise, if the approver is a customer, but has not approved the approvee.

Another possibility is that the approvee, or the page or message, was once approved, but that approval has now expired. This might be seen as not as bad as the previous cases, inasmuch as the approvee or item was once approved.

Another possibility is that the approvee, or the page or message, was once approved, but the approval was revoked. Here there could be several reasons, some of which are discussed below. Some of these reasons could be worse than others. Like if the approvee was discovered to be a phisher.

The browser's user might also be able to define how permissive the browser might be, if it finds that some Approval tags verify and others do not.

1.1 Extensions

In the above examples of approval tags, both the approver and approvee were specified in the tags. A variant on an approval tag might omit explicit mention of the approvee. Instead, the approvee might be derived from the context of the page or message. If we are looking at a web page, then the UKL gives us the approvee's

base domain. If we are looking at an email, then the sender field gives us the approvee's base domain.

This idea of multiple approvals could be used with any type of digital data. A document file might have approval tags embedded. So a program that can read the document can extract these tags and act in the manner of the browser plug-in, to test the document's approval. Or the file might be a binary file; i.e. a computer program. This insertion of approval tags into the binary file follows our idea in earlier Provisionals of using a tag in a binary file, where the tag was inserted by the author of the file.

We can also use the ideas in our Antispam Provisionals. In those was defined the idea of several metadata spaces that could be derived from a set of electronic messages. The spaces included domain, hash, style, relay and user. Here, we can define an "approval metadata space". This consists of mappings from approver to approvee. Where an approver can point to several approvees. And an approvee can have several approvers. Hence, one could make directed graphs ("clusters") to further investigate the relationships between these parties. Furthermore, these clusters have the virtue of being deterministic, because they are based on an objectively observed topological connectivity. By contrast, in data mining, clustering is often subjective. (Cf. "Introduction to Data Mining" by P Tan, Addison-Wesley 2005.)

This approval metadata can be combined with information from other Electronic Communication Modalities (ECMs) of "0046". For example, using the methods of "1745", an ISP or Aggregator might make clusters in various metadata spaces, from a corpus of email. Then, the Aggregator might combine a domain cluster derived from email with an approval graph or cluster. Giving rise to a cluster than spans two ECMs. The extra information given here, and the novelty of merging data from disparate sources, may let the Aggregator study the behavior of the approvees in a comprehensive manner.

Also, the Aggregator might share such information with its approver customers. Consider such a customer, charity5.org. It is approached by charityO, who would like charity5 to approve it. Presumably, the people in charity5 might then perform some manual scrutiny of charityO. But, charity5 might also ask the Aggregator for any metadata information about charityO. Where this could come from email data analyzed by the Aggregator or perhaps by its ISP customers. Suppose charityO appears in a domain cluster, where many other domains are considered to be spammers. Then, charity5 might decide that charityO is possibly a spammer, and hence declines to approve it.

This cluster investigation can also use more than just charityO's domain or network address. It might also look at any clusters that contain addresses in the neighborhood of charityO's address. For example, in the IPv4 addressing for the Internet, this might be a Class C set of addresses that contains charityO's address. Plus, we might also look at clusters with domains that have addresses close to charityO's address.

In addition to helping an approver approve or disapprove an approvee, the above steps can also be periodically done by the Aggregator or approver, in reviewing an approvee. Suppose that a month after charityO has been approved by charity5, the Aggregator compiles a domain cluster containing charityO, based on email. If the cluster has a "significant" number of spammer domains, or if the styles of the Bulk Message Envelopes ("0046") are considered undesirable, then the Aggregator might contact charity5 with its analysis. Here, the question of what constitutes a significant number of spammer domains might be where the Aggregator uses its expertise. Charity5 might also do its analysis. In any event, it may decide, based on the data, that charityO either is associated with undesirable others (spammers), or that charityO is a spammer. Initially, when charity5 approved charityO, this information might have been unavailable, and hence the approval was done. But now, charity5 might revoke its approval and tell the

Aggregator.

Also, charity5 or the Aggregator might inform the other approvers of charityO, so that they could review their approvals of charityO.

The Aggregator may also reserve the right to revoke one or more approvals of an approvee. Even if the approvers do not consent to this or have not (yet) been informed. While there might be several reasons, because the most important is that the Aggregator finds out from its own analysis, or is informed by others that it considers credible, that an approvee is bogus. Under these conditions, to protect others, the Aggregator might consider it imperative to immediately revoke the approvee's approvals. The Aggregator could also be compelled to take this action by a government regulator with jurisdiction over the Aggregator.

If an approval is changed or revoked, then the Aggregator might tell the approvee. Unless possibly the approver requested otherwise. Or if the Aggregator had a policy of not doing so.

The Aggregator can amass useful data from the incoming queries from plug-ins. For a given approvee, it can find any time dependence to the queries and any geographic dependence. Where the latter is inferred from the network addresses of the plug-ins. To be sure, the latter can be nullified if a user uses an anonymizer. But for most users, it might be reasonably expected that this will not occur. If the approvee has several pages or messages, then the Aggregator can do this for each of those, or for any subset.

The Aggregator can also study the relative effects of different approvers. This is information that individual approvers are unlikely to have. Each approver might only know about its own activities. The Aggregator may be able to use this comparative information to suggest to potential approvees, which are the most credible approvers.

If the Aggregator revokes an approval, as discussed above, and this is due to the approvee being considered fraudulent, then there are possible elaborations. The Aggregator might tell most plug-ins that ask about the approvee that the approval was revoked. Which protects those users against the approvee. But, the Aggregator might deliberately tell a few plug-ins that the page or message is approved. These plug-ins could be located in machines used by investigators, who might want to pretend to be fooled, and perhaps give the approvee false data.

1.2 Non-Charities

Thus far, we have discussed the context of a charity getting approvals from other parties. But our Invention has broader scope. It can be used by any website or author, that wants endorsements from other parties. Where the latter are assumed to be customers of an Aggregator, and hence presumably reputable and recognized authorities in their fields. While the approvee might be a startup company, or new author, that does not have name recognition, and wants to use our method to obtain some credibility.

Along these lines, an approver might charge an approvee for an approval. Similar to a company paying an accounting firm to audit its books. Or a company paying a bond ratings firm to analyze it, so that it can issue bonds.

Under these circumstances, the Aggregator might set a flag by such approvers or approvals, so that the plug-in can obtain it, and hence possibly indicate to the user that an approval was paid for by the approvee.

Whether endorsements are paid for or not, it has to be expected that an approvee will remove, or not place, any tags in its website or message that refer to an unfavorable "approval". The latter word is probably not the right term, in

general parlance. But we use it here, for consistency with the usage in the rest of this Invention, on the understanding that it can connote an unfavorable review. The Aggregator can offer these unfavorable approvals to the plug-in. So suppose the plug-in encounters a page with various approval tags. It goes to the Aggregator and finds that these verify. But it is also informed of other approvals that point to the page, but are not in the page. This can be especially useful to the user. So the plug-in might inform the user, in some fashion, that other approvals of the page exist, but are not referred to by the page. Hence, the user can read these. Plus, any metrics that might be found by combining data from approvals can, and perhaps should, include the data from these external approvals.

1.3 Search Engine

A search engine can also take advantage of our method. If it finds a page in its spidering with our custom tag, then it can perform the steps that a plug-in would do. If the verification fails, it could take action. Like possibly not making the page available as a search result. Because it might construe that the page is fraudulent, and hence protect its users by never giving the page.

It can expand on this, by perhaps scrutinizing other pages at that domain, and lowering their weightings. And, if several pages are found to have failed tags, then it might even remove all the domain's pages from its data.

However, if the tags verify, then it might choose to increase the weighting of the page. Since the page now has more credibility than a page without the tags. Which should improve the quality of the search results. And a page with several tags (that point to different approvers) that verify, might have a higher weighting than a page with only one tag that verifies, other factors being equal.

If the engine also has access to other types of data, like email, then it can

apply our method to them.

The engine can also offer various search options to its users, including but not limited to the following. To search only those pages that are approved. To search only those pages approved by 2 or more approvers. To search only those pages approved by a given approver.

Or course, the engine could also offer searching that involves the negations of the options in the previous paragraph, or any Boolean combination of those.

The approval network can be considered analogous to the network made by regular hyperlinks in web pages. Hence ideas used by the engine for ranking pages might also be applied here. One example is the use of anchor text. In a hyperlink, this is the visible text that appears between the <a> and tags. It is well known that an engine might use this, in order to help classify the page that is pointed to. (Cf. "Search Engine Marketing" by Moran and Hunt, IBM Press 2005.) The engine can programmatically record the anchor text, and use this, when finding results of a queiy. Similarly, if an approval has text, possibly in a comment, then this can be treated like anchor text.

1.4 User Feedback

Users with a browser and plug-in may be allowed to give feedback on a given page or message that has the approval tags. Typically, this feedback might be a simple "yes" or "no". The feedback could be communicated to the plug-in. Which could then transmit it to the Aggregator, who can then accumulate these from many plug-ins. The Aggregator could then summarize these and pass the results onto the various approvers. Giving them feedback as to what end users think of their approvals. The Aggregator might also do this for the approvees.

1.5 Alternative Tags

We described the use of a custom tag, with the example <approve ... />. This was chosen to be different from existing HTML tags, and well as common but non-HTML tags that might, for example, be used only by specific browsers. An alternative way to implement this Invention in a page or message is to expand the usage of the hyperlink tag, <a>. The standard notation for it is, for example,

 Click

Custom fields could be inserted, like this,

 Click

Here, the "from" attribute indicates an approver. There might also be a "to" attribute, to explicitly indicate the approvee. These fields are different from those in the official <a>, and hence a browser that does not have our plug-in will ignore them, and just show a standard hyperlink. But if the plug-in exists, then it can parse the <a> tags looking for the above custom attributes. If these exist, then it can contact the Aggregator and apply our method.

The plug-in might support both our <approve> tag and the above custom <a> tag.

If the custom <a> tag is used, and the plug-in is present and it computes a hash of the page or message, then it should first remove all such custom tags. Including the closing tags, and the visible text delineated by those tags. This lets the author add several custom tags, and yet keep the same hash.

1.6 Chain of Approvals

Above, we discussed the example of charityO being approved by charity5. Where charity5 is a customer of the Aggregator, and charityO is not. The Aggregator might then let charityO approve other companies. Suppose one of these is chess3.org. Then, a web page at that site, or a message from it, might have the tag

<approve from="charity0.org" to="chess3.org" />

The plug-in sees this, and does similar steps to earlier, sending a query to the Aggregator about an approver called charityO.org. The Aggregator sees that this is not one of its customers. But it then searches its customers' approval lists. It finds that for the customer charity5.org, it approves charityO.org. Hence, it can tell the plug-in that the approval exists. If so, it might also indicate to the plug-in that this is one degree of separation from its actual customers. This assumes that at some earlier time, when charityO found that it was approved by charity5, then charityO uploaded its approvals to the Aggregator, in the same manner that was done by charity5.

Continuing in this manner, the Aggregator might be willing to approve up to some maximum degree of separation. Hence we can construct a microtrust model that derives from a base of a set of companies that have been validated by the Aggregator.

Of course, the longer the chain of approvals, the greater the risk that some company in that chain is a fraudster. Which is why the plug-in may want to offer some signal about the length of the chain to the user. Or, the user might define a maximal chain that she will accept. If she gets to a page or message that has a longer chain, then the plug-in will automatically take whatever actions it would for an invalid approval. Naturally, the plug-in might offer a default maximal chain, to

make the user's choice easier.

1.7 Blacklists

The use of a blacklist against domains in hyperlinks of messages was described by us in "0046". If the blacklist has spammer domains, then this is a very simple and effective antispam measure. Now for the customers of the Aggregator, it might be unlikely, though not impossible, that they would use (or, rather, misuse) electronic messaging so that they would be considered spammers. However, if the Aggregator permits a chain of approvals, then entities further down the chains might be more likely to issue spam.

In either case, a message provider that gets incoming or outgoing messages can also apply the blacklist against domains in the approval tags. If an approver in such a tag is in the blacklist, then the provider might take the view that the approvee is also likely to be a spammer. The message can then be treated as spam. The provider might do this without checking the tag against the Aggregator. If the message is meant to be outgoing, then the provider might refuse to send it out, and might apply scrutiny to the sender, as a suspected spammer.

Note however that if the provider has a whitelist, and it finds a tag with the approver in the whitelist, then it should still check the tag. Because it has no assurance that the tag is in fact valid.

Clearly, if the approval data is written into the <a> tag, as discussed above, then the blacklist can be applied against the relevant custom fields in this tag.

If the message has scripting routines that dynamically make the approval tag, then this might be written by a spammer, to avoid an easy comparison with a blacklist. We discussed a related idea with standard hyperlinks in "1698". A simple

extension of that method can be used to handle any dynamic approval tags.

1.8 Web Services

Multiple approval tags can also be written into XML documents that are passed between servers running Web Services. A server might then have routines that perform tasks similar to that of the browser plug-in. These routines would extract the contents of an approval tag and check this against the Aggregator. These steps could be fully programmatic.

1.9 Advantages

Our method has several advantages over the current situation. The latter consists almost entirely, if not entirely, of manual steps that a casual user might be unaware of, or unwilling to perform. And even if she does do these steps, she can still make mistakes that cause her to be fooled by scam websites and messages. Those steps are mostly subjective. In contrast, our method is far stronger, because it can be done programmatically. and it is objective. Put simply, either a tag is right or it is wrong. It cannot be both. While we do permit shades of wrongness, there is still a strict demarcation between these and a tag that is correct.

Our method does not involve a parsing of keywords or any attempt to discern the "meaning" of a page or message. Which makes it language independent.

Plus, our method is computationally lightweight. There is no encryption involved. Except possibly to the extent that communication between the plug-in and the Aggregator might be done using https.

Also, since our plug-in is external to the page or message, it does not suffer

from the disadvantage of seals, which reside inside the page or message and can be faked. And which might still require manual effort on the part of the user, to see if the page or message is fake. If the item turns out to be real, then this false alarm will tend to reduce the user's inclination to perform such manual tests in the future, for other items. In our method, the manual effort needed by the user is mostly after a page or message is found to be fake. Which involves far less effort by the user.

Our method can be very germane to a government agency that regulates charities. It might act as an important approver, or it might combine the roles of an approver and an Aggregator. In the above discussion, for clarity, we have kept these roles separate. But a government may have the power to combine these.

Given the global reach of the Internet, our method can also be used by multinational agencies.

2. Verifying Blogging/Journaling

In the scope of this invention, we regard blogs, newsgroups and bulletin boards as fundamentally the same type of entity. We consider the cases of these which let visitors submit articles (and other content) in an unmoderated fashion. The articles can be considered "journaling". For brevity below, when we refer to a blog, this shall also include a newsgroup and bulletin board, unless otherwise specifically stated.

In our Antiphishing Provisionals, we described various elements that we now extend to tackle the above problem. Specifically, "5807" has a method of verifying links in web pages or messages. For a web page, it dealt with verifying all the links in that page. While for a message that is viewed in a program, like a browser, that can display hypertext, the method of "2528" lets us isolate the message body and hence only extract links from it.

One natural extension is to handle verifying different messages on the same web page. In general, these will be from different users. We can take the notation in Section 20 of "5807", where we presented these tags, <askLimit> and </askLimit>. They were used to delimit a section of an message. This section could then be hashed by the Asker. Within the section would be a tag <ask link="10.20.30.40:301" ...> (for example). Also in the tag might preferably be an id of the author. The Asker could then go to that network address and ask a Checker process, which is assumed to be present there, for verification about the message. The Asker might submit the (id, hash) and the Checker checks if the id exists, and if the hash is associated with the id. Thus far, in this paragraph, we have described strictly steps in "5807".

(The above could also be simplified by merging the fields of the <ask> into the <askLimit>. While we will not assume this in the further discussion, it is another possible implementation.)

We now make these extensions. The web page shows a blog. Whereas in "5807", the page was made by a message provider, in which the user is reading a message (like email). Next, instead of a single message being viewed in a browser, it is replaced by one or more articles. These could be from different visitors. Typically, a visitor writes an article by pressing some link or button in the page, which brings up another page where she can type. Or the first page might have a box where she can directly type. Or she might be able to email the article to the blog site.

The next change is that an article might not have any standard hyperlinks. In "5807", it mostly discussed the case of verifying such links. Here we handle the case where these are absent. But if the article has the <askLimit> and </askLimit> tags, and the <ask> tag between them, then a modification of the Asker and Checker of "5807" is possible.

"5807" also described various forms that a Checker could take. One variant is that the author ("Jane") of an article writes the above tags, and one of these points to another website. In which she then registers the hash of the article with that website's Checker, where the hash is made of the material between the <askLimit> and </askLimit> tags. The Checker stores a map from an id assigned to her to a list of such hashes. With possibly other information supplied by her. Hence it could answer a query from an Asker.

Alternatively, a simple form of a Checker could merely be a web page written by Jane, in which she posts the hashes.

In either case, she might deliberately omit putting in the Checker the address of the blog page where she wrote her article. This means that someone else who visits the page might copy her article and then post it, unmodified, at a different website. This might be ok to her. So that she does not need to add the blog's address to her Partner List (PL) at the Checker. Indeed, she might choose not to maintain a PL, if all her articles at various blogs only need to be validated for content, and she does not care about copying.

It also means that she might preserve some level of obscurity, with regard to which websites she writes articles in. The Checker only knows the hashes. It does not know the web pages these refer to. Whereas in "5807", the PL held at the Checker immediately gives it that information. Of course, here, as the Checker gets queries about Jane, with hashes that verify, then it can choose to build up a list of such pages over time. Jane cannot prevent this.

Many of the scenarios discussed in "5807" concerned a web page or message where only Jane could write to it. Thus, it was usually sufficient to just have a Checker verify links in what Jane wrote. But now we have the possibility that several people can write to the same page. This allows an attack where a spammer sees a verified entry written by Jane. Then, the spammer submits another article,

with different text and mostly different links, and having one other link be the same as a verified link in Jane's article. Thus, if just links are verified, the spammer can give the correct appearance that one of the links is verified, even if the others are not. Offers partial credibility to the spammer's article. But if Jane hashes her text, it blocks that attack. All the spammer might be able to do is duplicate Jane's article. Which is fairly harmless.

Given the presence of a network address inside the custom tag, the blog site might apply a blacklist against such addresses. So that if an address was in the blacklist, the blog might take action, like deleting the message. The blog might also check if that network address was within some neighbourhood of an address in the blacklist. For example, if it is in the same Class C neighbourhood, under IPv4 addressing. If so, then the blog might have a policy that this is tantamount to the message being undesirable, and hence delete it.

We also extend our anti-"Man in the Middle" method of "5809". In it, we described how a user taking her computer to an unfamiliar location, and getting a temporary address, could face a MITM, run by a phisher ("Amy"). Where Amy's machine sits between Jane's machine and the bank that Jane wants to login to. We offered an easy technique of Jane hashing a combination of her password and her temporary network address. And then passing this plus her username at the bank, to the supposed bank, which is really Amy. Thus, it does no good for Amy to pretend to be Jane, since in general, Amy will be seen by the bank to be at a different network address than Jane. In "5809", the hashing is of a secret and Jane's address.

Now, in this Invention, Jane can choose to find a hash of the combination of (the address of the blog page in which her article appears) PLUS (the text of her article). Here, all the information that goes into the hash is public. Jane then submits the resultant hash to her Checker. In the <askLimit> or <ask> or </askLimit>, she adds an attribute indicating that the verification hashing should

also include the page address. Optionally, extra parameters might be written to the tag. For instance, indicating which hash method was used. If this is omitted, then an implementation might have promulgated some default choice of hash method.

In this usage of the address, she can decide to specify only a subset of the address. Letting the blog owner have some leeway in redeploying the article at a different address within that address range. For example, suppose the page is at

http://ape.blog.info/dog/bird/goat.html.

She might hash only "http://ape.blog.info/dog/bird/". Hence the blog owner can move her article to any file in that directory. Or, Jane might hash "blog.info". So her article can sit at any address with that base domain. Or, suppose the domain maps to the raw address 20.30.40.50, and the blogger owns the range 20.30.40.*. Then Jane might hash "20.30.40.*". So that her article can sit anywhere in this raw range. (Here, we use the Internet Protocol version 4 addressing. But our remarks also hold under IPv6, with the obvious generalizations in notation.)

Whichever her decision, this choice of which fields to hash can also be encoded into the tag. So that a visitor knows what to hash when verifying.

When we said Jane makes this choice, it might actually be on advice by the blog owner. Who might require or suggest that if an author wants to make her article verifiable and include address information, then she should only use certain fields. Perhaps to give the owner leeway to move around the pages. This advice from the owner to Jane can be done in a programmatic manner. For example, imagine the owner exposing a Web Service that Jane's browser can detect. The Service then specifies the addressing constraints and whether these are mandatory or optional. The browser can use these to automatically find the hash, when Jane is finished writing the article. Or, in the page where Jane submits her article, there might be these hashing options. And the blog site then finds the hash, according to

these choices.

It should be understood that when we said hash the address+text, that this can be taken to mean hashing some mathematical combination of the two variables. This parallels the discussion in "5809". So, for example, our method includes hashing in this order the text+address. What matters is that the steps done by Jane are the same as those done by a visitor to the blog, that is attempting to verify her article.

Hence, Jane locks down the verification to that web page. Anyone copying her article will not be able to have it verified at a different address. This locking is an alternative to the use of a PL in "5807". In the Web, there is no obstacle to a user with a browser copying a typical web page, and putting that copy at another address. Indeed, this has been one of the factors driving the Web's growth. But though copying cannot be prevented, verification can.

(The previous paragraph has a minor caveat. Some web pages might have intricate functionality that preclude a simple copy. But most pages, and this includes most blog, newsgroup or bulletin websites, can indeed be easily copied. Or subsets of those pages, like single articles.)

As described in "5807", the website (blog) where Jane (and presumably others) wrote articles can offer a verification ability. For example, it might highlight in some manner the articles that have been verified. While unverified articles (which lack the tags) are shown in another fashion. And invalidated articles (these have tags, but are found to be invalid) are shown differently. Standard display options might be to only show verified articles, or to only show unverified articles.

2.1 Bot Articles

Now suppose a bot visits the blog and writes an article. If it lacks our tags, then the blog owner has an extra programmatic heuristic of finding such articles. The problem still exists of being being able to distinguishing between the bot article and regular articles that don't have our tags. The blog can use blacklists of spam sites, and apply these against any links in articles. For spam sites that are long lived, and which employ bots to write blog articles, a comprehensive blacklist can help a blog detect many of these articles.

Suppose the bot article has our tags. If these do not verify, then it is a very strong signal that the article is dubious.

Hence imagine that the bot article has our tags and it verifies. The blog may have a special whitelist of Checker domains. These are domains which it considers to be reputable and which have policies against verifying spammers or users for which there have been substantial such complaints. If the bot article verifies against a Checker not on the blog's Checker whitelist, then the article might be deleted.

Plus, of course, the blog can still apply its spam blacklist against any links in the verified articles. If it finds such an article, then this can be used as the basis for a complaint to that Checker about the article's author.

2.2 Search Engine

Our method can also be used by a search engine ("Engine") that spiders the blog. It might spider blogs, simply because these are publicly accessible web pages, and so become part of the Engine's scope. A problem that has emerged with blogs is "search spam". These bot articles are often written by a program or a human that visits the blog and submits what is really an ad for a good or service.

Sometimes, conceivably, the owner of the blog might insert such articles. Perhaps one reason for the owner to maintain the blog is to be able to sell such ads. Like a lot of email spam, search spam often has links to, websites offering goods or services. The spam articles might be not just for humans to read and click on the links. They might also be to skew the Engine's rankings of websites. If the Engine spiders many blogs, and finds many spam articles pointing to a given website, then that website might rise in the Engine's rankings. Undesirable for two reasons. Firstly, this ranking does not reflect true, independent assessments by different websites of that website's popularity. The value of the search results is diminished for users. Secondly, the spammer does this to avoid buying ads with the Engine. The Engine loses revenue.

Now, the Engine can give higher weighting to blog articles that can be verified using our method. It can do this, independently of whether the blog does so or not. This is to account for the case where the blog owner is responsible for the spam articles, and might falsely claim that the articles are verified. Also, the Engine can detect this false claim. If so, then it has extra information about the blog site. It might consider the site to be highly suspect, and deprecate the site's weighting or even drop the site from its survey.

2.3 Non-Text Verifying

Suppose Jane uploads an image file into her article, or an audio file, or a video file, or some other type of data. Her tag can indicate whether these should be be added to the text of her article, in order to hash the combination. Or, if these files should be hashed separately. In this case, an elaboration is for a hash to be made of the combination of (text+file hashes). So as to produce a final hash that binds the entire contents of the article together. With the address of the page also added to the input to the hashing, as discussed above, if Jane want to bind to the address.

The tag notation can be generalized to indicate which of the assets in her article should be hashed. She may want this precision, so that, for example, she can state that only the text and images should be hashed, while audio files can be skipped.

For a visitor with a browser going to the page with Jane's article, finding the hashes is a little more involved than above. The assets which are immediately available to the visitor are the text and any images. But for an arbitrary file in the article, it may need to be downloaded to the browser's memory, for hashing to be done.

The previous paragraph is moot to the blog, if it wishes to verify Jane's article. Since Jane has, by assumption, uploaded all the assets of the article to the blog's computer.

2.4 Aggregating Articles

A popular trend is for a newsfeed to be aggregated from multiple sources. Perhaps using the RSS (Really Simple Syndication) methods. One possibility is that Jane might have her own blog site, where she writes her articles. She might then want some other sites to use her articles. One way, under RSS, is for her to supply an RSS news feed. While is basically a list (file) written in XML, with links to her web pages, and possibly to specific articles on a given page. By publishing this list on the Web, other sites could then link to her articles. Another approach is for those sites to copy (hopefully verbatim) her articles. As we've mentioned earlier, this copying is essentially impossible to prevent. But Jane can put restrictions on which websites her articles will verify on.

Assume that when she wrote her article for her website, she just did a hash of

its text, and did not include her website address in the hash input. In the Checker, she can supply ancillary data to be associated with this hash. It could be a whitelist of addresses at which she approves, to physically hold a copy of her article. This entries in the whitelist might be full addresses. Or, perhaps more realistically, they might be base domains. So an entry of, say, ballbat.com, means that the Checker should say that any request about her (id, hash) at an address with that base domain, is verified.

Jane might also have a blacklist of addresses which will not verify. A question arises that having both a whitelist and a blacklist seems redundant? Possibly. Jane might have only a whitelist, and a policy at the Checker that if an address is not on the list, then the article is not verified. Or she might have only a blacklist, and a policy that if an address is not on the list, then the article is verified.

Either list might have the ability to have wildcards. So that, for example, a whitelist entry of "*.gov" means that any address in the US government domain is considered good.

The Checker might supply a default whitelist and blacklist, so that its customers can use these in conjunction with, or perhaps instead of, their own lists.

2.5 Provenance of Repeated Aggregation

Suppose we have a blog site, Theta.com, that collects articles via newsfeeds (like RSS perhaps) and possibly also by contributors directly writing to it. Imagine another blog site, Phi.com, that does likewise. And Phi takes articles from Theta.

Theta can essentially act as the "author" of the articles posted on its site. For example, with a given article, it might encapsulate that with <askLimit> and

</askLimit> tags. Even if that article already has these, from its existing, actual author. Then, Theta can add an <ask> tag. Where, to reduce chances of ambiguity with any existing <ask> tags, Theta's tag goes outside the scope of any internal <askLimit> or </askLimit> tags.

(Or, when any <askLimit> and </askLimit> tags are used, there might be no <ask>. Instead, as mentioned earlier, the <ask> fields are put into <askLimit>. Which simplifies the above encapsulation.)

If Theta does this, it might run its own Checker, or refers to some other Checker. In either case, it registers its id and hashes with the Checker. (If the Checker is its own, then the id is probably superfluous, if this Checker only verifies Theta's data.) Hence, a visitor to Phi's website can use the method of the earlier sections to see that the articles from Theta are verified. This is true whether or not the articles have any verification from other Checkers. If they do, then our method can be easily extended to let the visitor's browser (or Phi) perform verification using those other Checkers.

In this fashion, a visitor can see the provenance of an article, as it propagated through the blogs or newsgroups. Even if the original owner is Anonymous, being able to verify the trajectory of the article may have value to some visitors. Or to a blog that might have policies about what articles it will accept. Imagine that Phi might not accept any articles submitted directly to it by an author, that it cannot verify. But it will do so, if these come from Theta. Phi can do this programmatically, which is a significant saving over using manual effort. In this example, maybe Theta accepts unverified articles. But, say, it expends manual effort to somehow check these. In general, Theta's cost for doing this will not be passed in its entirety to Phi. Or perhaps even at all, if Phi gets a free feed from Theta.

Suppose Phi shows articles from many sources. It might have logic that uses

some type of reputation service (which may be external to it) to make decisions like that in the previous paragraph.

Of course, Phi does not have to abide by our method. When it displays an article that comes from Theta, it might strip off any enclosing tags. Or even all such tags, throughout the article, which wipes out any verification ability to the visitor. But an attraction of our method is that it lets websites that use it compete with websites that do not. Not all websites (and their visitors) will see a need for our method. Others might. Our method allows for incremental adoption by authors and websites. And the articles that are generated with our tags are compatible with websites that do not use these.

Suppose we have an article that has traveled through various blogs, with each encapsulating it in the blog's verification tags. When a blog gets this article, it can keep those tags. But suppose it makes changes to some. Or it deletes some, but not all. Or it inserts fake tags. If it presents the article as verifiable, then these can be detected, unless it chooses to delete all the valid tags. By the encapsulation and the hashing of everything inside a corresponding pair of tags, this acts to bind the trajectory and the content. Actions by a rogue blog will cause the verification to fail.

We can compare this to the situation in email. Each email relay adds its header information to a message that it gets and sends on. But the email protocol was written at a time when there was no reason why a relay might enter false information. Or why a message going to a relay might have false information about the message's purported route that that relay. Spammers have taken full advantage of this. One proposed antidote is for digital signing of messages, including their headers, as they go through the Internet. This has proved difficult to implement for several reasons, including issues of key distribution and the increased complexity of handling the mail.

In contrast, our method involves hashing and not signing. There are no keys to distribute and maintain. And the computational effort in hashing is less than in signing. Plus, there are fewer hashes than there would be signatures, assuming that digital signing was used for most email. Each day, several billion emails are sent out, spam and non-spam. But not several billion new articles for blogs and the like. Even where a program might generate these, like spam, there is no incentive to ramp up to the numbers seen for spam. The bottleneck is the blog pages. Writing the same spam thousands of times to a given blog page is very unlikely to significantly increase the chances of readers buying its offering.

2.6 Anti-MITM Extension

Above, we described an extension to the anti-MITM method of "5809". Here, we describe another extension. This involves the following topology. A user, Laura, has a mobile computer, which she connects to the network, via another party, Amy. Laura wants to login to her bank account at bank0.com -

Laura <----------> Amy <----------> bankO.com

Amy represents herself to Laura as various possible legitimate guises. Perhaps as a WiFi connection. Or as the network connection in a cybercafe or public library. But unbeknownst to Laura, when she tries to connect to bank0.com, she is really going to a fake site, that possibly has BankO's real network address. This is where Amy might have a false DNS mapping for bankO, or in other fashions, misrouting Laura's message. In "5809", we described how if Laura is allocated a network address that can be seen by the rest of the network, then she can include it in the input to a hash. The latter is sent to bankO. Which can then verify it independently, by seeing what address the (apparent) Laura is at.

Another possibility is that when Laura connects to what she thinks is the

network, that explicitly, Amy might be running what amounts to a dynamic DNS. For example, Amy might only have one address facing the outside network, which might and probably will be the Internet. She runs Network Address Translation (NAT). Users like Laura get a temporary address that cannot be seen on the outside network. When Laura goes to an arbitrary address on the outside network, the NAT converts her address into Amy's outside address and some type of id, unique to Laura's session. For example, a cybercafe might do this, with no malign intent, if it only has one outward facing address.

Suppose though that Amy does the steps in the previous paragraph, and also redirects Laura's queries to bankO.com to an outside website run by Amy. Which then forwards Laura's messages in a MITM manner, to the real banlcO.com. As in "5809", Laura and bankO have a common shared secret (her password). Under these conditions, Laura and the bank could implement the following method.

She picks an integer, i, and makes a hash of a combination of i and her password. She sends (i, hash, username) to what she thinks is bankO.com, but which is really Amy. Who then forwards a copy to the real BankO. It verifies (i, hash, username). Then it and Laura use i and her password in some fashion to derive a key to encrypting messages between them. Hence, even though Amy will get these encrypted messages, she won't be able to decrypt them.

CLAIMS

1. A method where an author of a web page, with address Theta, can add a custom "Approval" tag, with an attribute that designates the address of another network location ("approver") that approves or endorses Theta.

2. A method, using claim 1, where there could be several such Approval tags in Theta, each tag referring to a different approver.

3. A method, using claim 1, where there is a central website, Agg, that maintains a set of approvers, that it approves of; where each approver can submit to the Agg a list of addresses of pages at other websites that the approver approves; where with each address in that list might be a list of base domains in links in the approved page, or a hash of the approved page.

4. A method, using claim 3, where there is a browser modification, such that when the browser detects an Approval tag in a page it is displaying, it contacts the Agg to obtain information to verify the tag; where this verification might involve finding base domains of links in the page, or of finding a hash of the page or of a subset of the page, and comparing such information with that from the Agg.

5. A method, using claim 4, where if the Approval tag does not verify, then the browser indicates this in some manner to its user, including, but not limited to, turning off any links in the page being displayed, or not displaying the page.

6. A method, using claim 4, where if the Approval tag does verify, then the browser indicates this in some manner to its user.

7. A method, using claim 3, where the approver can set an expiration date on a given address that it approves of.

8. A method, using claim 3, where the approver can specify a domain of an approvee, meaning that it approves of all pages within that domain, including any subdomains.

9. A method, using claim 8, where the browser can apply this data to the approvee's pages.

10. A method, using claim 3, where an approver indicates that an approval of an approvee's page was paid for by the approvee.

11. A method, using claim 10, where if such a page's Approval tag verifies, then the browser can indicate in some manner that the approval was paid for.

12. A method, using claim 4, where if a page has several Approval tags, and some verify and some do not verify, that the browser has logic to indicate these "partial" verifications to the user in some manner.

13. A method, using claim 12, where the user can set a browser policy as to how strict the verifications must be.

14. A method, using claim 3, where the Agg might reject an approvee, perhaps based on a determination that the approvee is a spammer or phisher.

15. A method, using claim 3, where a search engine, S, spidering a page with an Approval tag, verifies it with the Agg.

16. A method, using claim 15, where S increases the weight of a page that verifies, and decreases the weight of a page that does not verify, in determining search results; and possibly does not make the latter pages accessible in search results.

17. A method, using claim 15, where S lets a user restrict a search to pages with Approval tags that verify.

18. A method, using claim I_5 where a message provider can apply a blacklist against an approver in an Approval tag in an incoming or outgoing message; and if the approver is in the blacklist, the provider can take various steps, including possibly marking the message as spam.

19. A method, using claim 4, where the user can vote on a page with an Approval, and the browser sends this information to the Agg, which can aggregate such votes per page and make results available to the approver or approvee.

20. A method where the writing of a message in a page of a newsgroup or blog might include custom tags, that delineate the message within the page, and which have an address of another network location that approves of that message; and where the page might have messages written by several authors.

21. A method, using claim 20, where the blog might have code that detects such messages with those custom tags, and verifies them by querying processes ("Checkers") at those other network locations.

22. A method, using claim 20, where a search engine spiders such a page, and verifies any messages with those custom tags, and uses the results to modify the weight of the page, when determining search results.

23. A method, using claim 20, where the blog might apply a blacklist against a network address inside the custom tag, and if the address is in the blacklist, then the blog might delete the message.

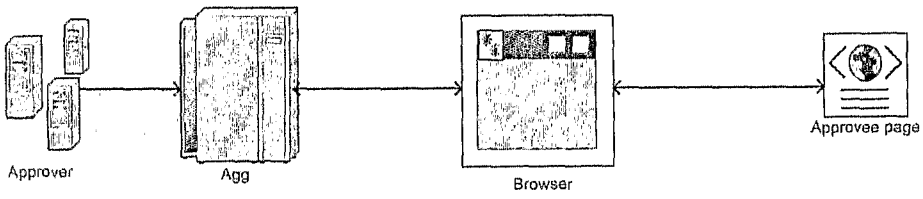


FIG. 1

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CN2006/003728

A. CLASSIFICATION OF SUBJECT MATTER

See extra sheet

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: H04L, G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPI, EPODOC, PAJ, CNPAT, CNKI : WEB W PAGE TAG OR TAB OR LABEL APPROV+ OR THE SIMILAR

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
PX	WO200602692 1A2 (METASWARM HONGKONG LTD) 16 Mar. 2006 (16. 03. 2006) See the whole document	1-23
PX	JP2006313517A (E-LOCK CORP SDN BHD) 16 Nov. 2006 (16. 11. 2006) See the whole document	1-23
X	US5937404A (APPALOOS A INTERACTIVE CORP) 10 Aug. 1999 (10. 08. 1999) See column 3 line 5-25, column 5 line 35-55 in the description, fig. 3	1-23
A	US2004199606A1 (INT BUSINESS MACHINES CORP) 07 Oct. 2004 (07. 10. 2004) See the whole document	1-23

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim (S) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&"document member of the same patent family

Date of the actual completion of the international search
20 Apr. 2007 (20. 04. 2007)

Date of mailing of the international search report
17 MAY 2007 (17.05.2007)

Name and mailing address of the ISA/CN
The State Intellectual Property Office, the P.R.China
6 Xitucheng Rd., Jimen Bridge, Haidian District, Beijing, China 100088
Facsimile No. 86-10-62019451

Authorized officer
WANG Zhiwei

Telephone No. 86-10-62086080

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2006/003728

CLASSIFICATION OF SUBJECT MATTER

G06F11/28 (2007.01) i

G06F15/16 (2007.01) n

H04L9/00 (2007.01) n

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No,
PCT/CN2006/003728

Patent Documents referred in the Report	Publication Date	Patent Family	Publication Date
WO2006026921 A2	16.03.2006	NONE	
JP2006313517 A	16.11.2006	US2006253446 A	09. 11. 2006
		AU200620068S A	23. 11. 2006
US5937404 A	10. 08. 1999	WO9848546A	29. 10. 1998
		CA2287341 A	29. 10. 1998
		AU6977598 A	13. 11. 1998
		EP0978185 A	09. 02. 2000
		AU719455B	11. 05. 2000
		JP2001503178T	06. 03. 2001
US2004199606 A 1	07. 10. 2004	NONE	

Form PCT/ISA/210 (patent family annex) (April 2005)