Design and Implementation of the Lucent Personalized Web Assistant (LPWA)

David M. Kristol Eran Gabber Phillip B. Gibbons
Yossi Matias* Alain Mayer

Information Sciences Research Center
Bell Laboratories, Lucent Technologies
600 Mountain Avenue
Murray Hill, NJ 07974
{dmk, eran, gibbons, matias, alain}@research.bell-labs.com

June 10, 1998

Abstract

This paper describes the design and implementation of the Lucent Personalized Web Assistant (LPWA). LPWA is a software system that enables a user to browse the Web in a *personalized*, *simple*, *private*, and *secure* fashion and to filter junk e-mail (spam).

LPWA generates secure, consistent and pseudonymous aliases (personae) for Web users. Each alias consists of an alias username, an alias password and an alias e-mail address. The alias e-mail addresses allow web-sites to send messages to users and enable effective filtering of junk e-mail (spam). LPWA forwards mail addressed to the alias e-mail address to the actual user. LPWA allows users to filter incoming messages based on the recipient address (the alias e-mail), which is an effective method for detecting and blocking spam.

A trial version of LPWA became available to the public at `http://lpwa.com` in June, 1997. It has so far (as of May, 1998) attracted more than 21000 users.

1 Introduction

In recent years the World-Wide Web (WWW) has become an immensely popular and powerful medium. Easy access to a large variety of information has attracted many users. To attract more users, many web-sites offer *personalized service*. For example, news sites, such as `my.yahoo.com`, `my.excite.com` and `www.news.com`, let users register their preferences for news topics, stock quotes, weather reports, etc. On return visits, the user is conveniently presented with the chosen selection of information.

On the other hand, personalized services raise user concerns with respect to convenience and privacy. Registration for these services lets information providers use a variety of tools to collect extensive profiles of users who visit their web-sites. Moreover, registering typically requires the user

*Also with Computer Science Dept., Tel-Aviv University, Tel-Aviv 69978 Israel. E-mail: matias@math.tau.ac.il.

to specify a unique username and a secret password. Upon each return visit, the user must provide the same username and password. Sound security would dictate that users choose (and remember!) a different password for each site. An additional problem arises when untrained users choose the same password for a web-site (e.g., for `my.yahoo.com`) as they use for their own company's machines, thus potentially providing an intruder an easy way to break into the company's intranet.

Many sites ask for an e-mail address at registration time as well, which can effectively serve as a (nearly) unique identifier for a user and which thus provides an avenue for profile aggregation across web-sites. Furthermore, a database of user e-mail addresses can be easily abused to send out junk e-mail (spam). To counter these concerns, users either avoid sites that require them to register, or they register with false information. However, an increasing number of web-sites use the supplied e-mail address to send back a verification number needed for return visits. Some web-sites offer periodic mailings. For example, the travel site `expedia.com` e-mails best fares for user preferred airline routes. Hence, the user often must supply a valid e-mail address to use a service at all.

This paper describes the *Lucent Personalized Web Assistant (LPWA)*, a novel software system designed to address these user concerns. Users may browse the Web in a *personalized*, *simple*, *private*, and *secure* fashion using LPWA-generated aliases and other LPWA features. LPWA provides the following functionality:

- *Automatic, Secure, Consistent and Pseudonymous Generation of Aliases:* Aliases present a different persona (username, password, e-mail address) to each web-site. Personae for different web-sites, but belonging to the same user, appear to be independent and unrelated. (We will use "persona" and "alias" interchangeably in the rest of the paper.) The generated aliases are consistent, which means that the user will present the same alias on return visits to the same web-site. They are pseudonymous in the sense that one cannot correlate between different aliases of the same user, nor between a user and its aliases.

- *E-mail Service:* Web-sites can use the e-mail address of the supplied persona to send information to the user.

- *Anti-Spamming Support:* Users can filter junk e-mail based on the *recipient e-mail address*, which happens to be the persona e-mail address. Furthermore, the user can infer which web-site is responsible for compromising the e-mail address, even when the message is sent by a third party.

- *Filtering of Privacy-sensitive HTTP Header fields.*

- *Indirection:* The TCP connection between the user and the web-site passes through a proxy, which thwarts tracking of the originating computer.

In a companion paper [BGGMM98], we discuss the basic notion of pseudonymous client-server schemes. We introduce the *Janus function* $\mathcal{J}$, which provides a client with a different persona for each server. The companion paper also gives a cryptographic design of $\mathcal{J}$ and shows a particularly elegant idea of an e-mail storage and retrieval scheme that requires no storage of privacy-compromising information.

LPWA is a particular instantiation of a pseudonymous client-server scheme, customized for the Web. Furthermore, our goal was to build a system that could be readily deployed for public use. In the rest of this paper, we will present the design and implementation of the public trial version of LPWA. We will discuss the particular difficulties we encountered and the compromises we chose.

2 Overview of LPWA

LPWA has the following three functional components:

- *Persona Generator*: Generates a unique, consistent site-specific persona on demand by a user. The generator requires two pieces of identity information from a user: a *User ID*, which is a valid Internet e-mail address for the user; and a *Secret*, which serves as a universal password. Using these two pieces of information, plus the destination web-site address, the generator computes a persona for this web-site on the user's behalf.
- *Browsing Proxy*: Increases the user's privacy by indirecting the connection on the TCP level and filtering headers on the HTTP level.
- *E-mail Forwarder*: Forwards mail, addressed to a persona e-mail address, to the corresponding user.

LPWA's functional components can potentially reside at various places. The Persona Generator can be implemented directly within the user's browser or on the Browsing Proxy. The Browsing Proxy might reside on a firewall, an ISP access point, or a neutral site on the Internet. The E-mail Forwarder needs to reside "away from" the user's machine, since the goal is that the various persona e-mail addresses would be unlinkable to the user. Obviously, there are various trade-offs involved:

- *Trust:* The Persona Generator receives the user's real e-mail address and a secret. The user opens a direct TCP connection to the Browsing Proxy. Depending on the design, the E-mail Forwarder must reliably either store or forward the received messages. Hence, all components must be trusted to various degrees.
- *Anonymity:* Neither the Browsing Proxy's nor the E-mail Forwarder's location should make it possible to infer a user's identity.
- *Performance:* If the location of the Browsing Proxy is "too far away" (in terms of Internet connections), then the performance degradation when browsing becomes noticeable to the user. This is an inherent problem of all HTTP proxies, since all traffic to and from the user's browser is routed through the proxy.
- *Ease of a Public Trial:* Certain issues become relevant in this context, such as distribution of source code containing cryptography or the availability of browser source code.

Our intention to quickly deploy a trial version prevented us from considering browser changes (no source code was available at the time). Furthermore, distributing software that contains cryptographic modules posed difficulties that at the very least would delay our trial considerably. As a result, we decided to implement LPWA for a public trial using the following two components:

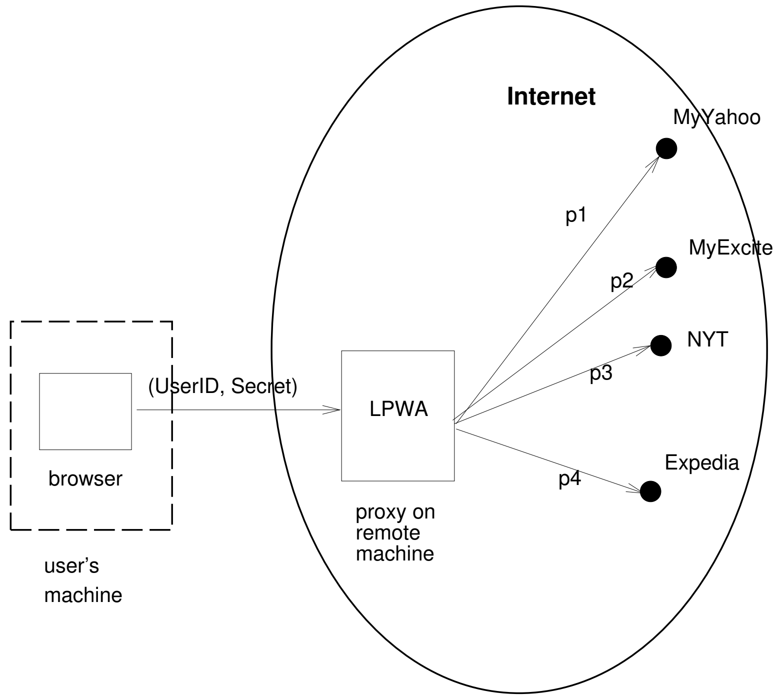

Figure 1: LPWA HTTP proxy configuration

- An HTTP proxy server, located on our premises in Murray Hill, New Jersey, that implements both the Browsing Proxy and the Persona Generator. This configuration is depicted in Figure 1.
- A remailer, located on the same machine as the proxy server, that implements the E-mail Forwarder.

In [BGGMM98], we discuss schemes with components residing on user's machines, ISP access points, or firewalls. Compared to our choice, such configurations have advantages in terms of trust and performance, as discussed in detail in [BGGMM98]. On the other hand, our design choices allowed a fast deployment of a public trial version, showcasing our ideas and attracting thousands of users; see Section 8. (We have also implemented an internal trial version, for users within the Lucent corporate firewall; this version plays the role of a firewall proxy.)

2.1 Usage of LPWA

This section summarizes a user's interaction with LPWA. Further details will be provided in subsequent sections.

The user configures her browser's HTTP Proxy setting to use the LPWA HTTP proxy. (The current trial LPWA proxy is located at `lpwa.com` .) Subsequently, at the beginning of a browsing session, the user is presented with the LPWA start-up page, as depicted in Figure 2 (the user can use the quick login on the right hand side, or the safer, more elaborate, login on the left hand side). This page asks the user to supply her User ID (real e-mail address) and Secret (universal password).

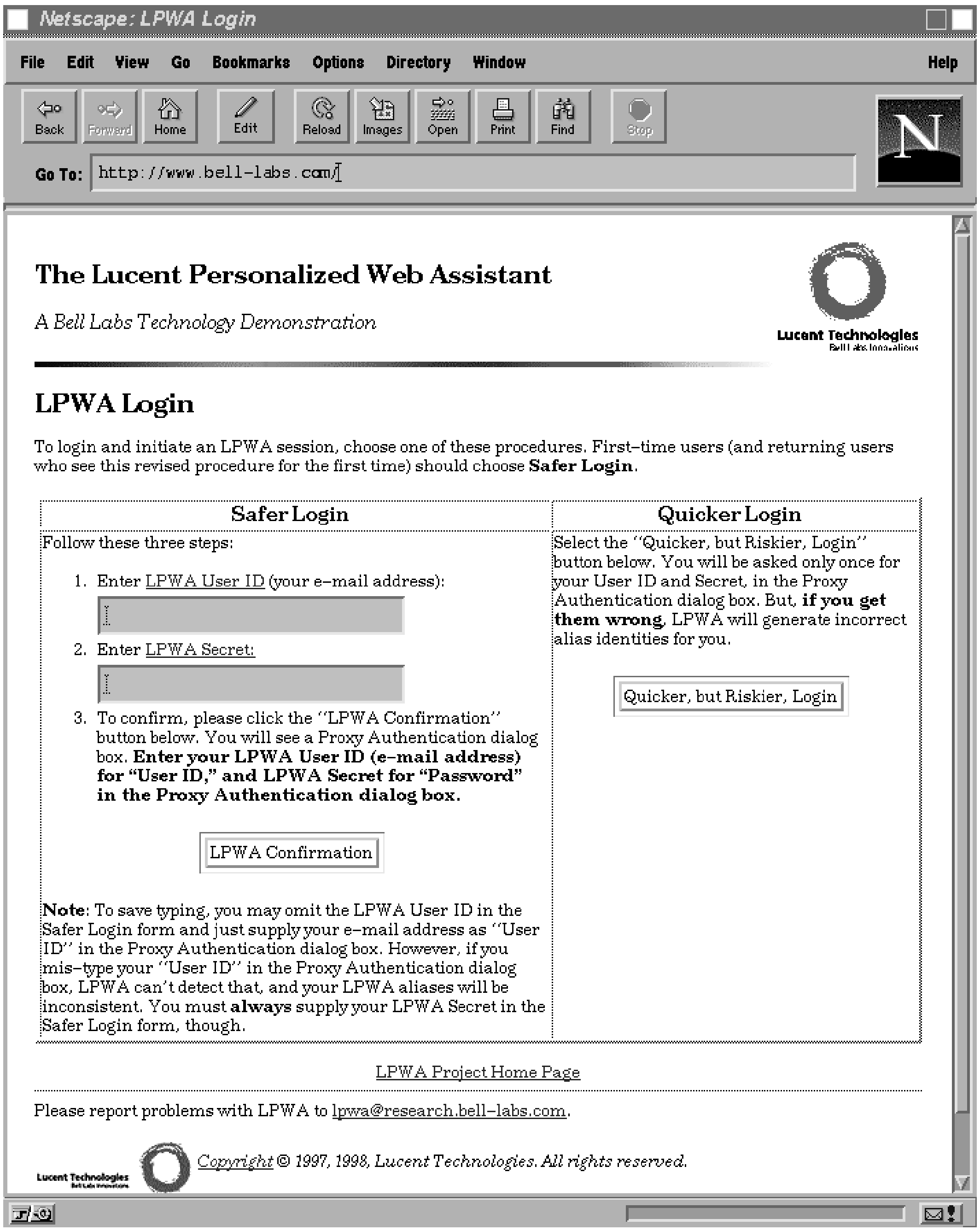

Figure 2: LPWA start-up page

From that point on, LPWA is transparent while the user is browsing the Web. Whenever a web-site asks the user to supply any of her username, password, or e-mail address, the user may invoke LPWA by supplying a corresponding LPWA escape sequence, as depicted in Figure 3 for the New York Times web-site. As it passes along the request to the destination web-site, LPWA recognizes these sequences, computes a persona username, password, or e-mail address specific to that web-site, and inserts them into the user's request. On repeat visits, LPWA will produce those same personae, so when the user returns to a web-site, she is recognized as a repeat visitor. When a web-site sends a message to a persona e-mail address, the message arrives at LPWA, which then forwards the message to the corresponding user.

3 Design of the LPWA HTTP Proxy

In this section, we first present our design requirements for the proxy. Then we show how the user supplied identification (User ID and Secret, as described in Section 2) is managed. Finally, we describe LPWA's filtering of privacy-sensitive HTTP header fields.

3.1 Requirements

We had several design requirements for the LPWA HTTP proxy:

1. It should work with most already-available web browsers.
2. It should be easy to use and should work transparently.
3. It should be relatively easy to implement.
4. It should be stateless.
5. It should follow the HTTP standard, RFC 2068 [RFC2068].

Browsers: We wanted to be able to build, test, and deploy LPWA quickly. These factors precluded any kind of custom web browser. Thus although the LPWA technology could be incorporated into a web browser, we deliberately chose a mechanism that would work with existing browsers.

Easy to Use and Transparent: If users were going to find LPWA convenient, it had to be easy to use and non-intrusive. So we made it simple to set up a browser to use LPWA, and, after the initial identification, LPWA is invisible. Moreover, the user does not have to download any software, and the browser's setup is performed only once.

Easy to Implement: We decided to base the LPWA proxy on the Apache Server, a public domain server produced by the Apache Group. This server is widely used, and the source code is freely available and actively supported. We found that our changes could be inserted "surgically" with modest changes to the existing code base.

Stateless: For both operational and privacy reasons we decided that the proxy server should retain no information about user identities. From an operational standpoint, making the server stateless

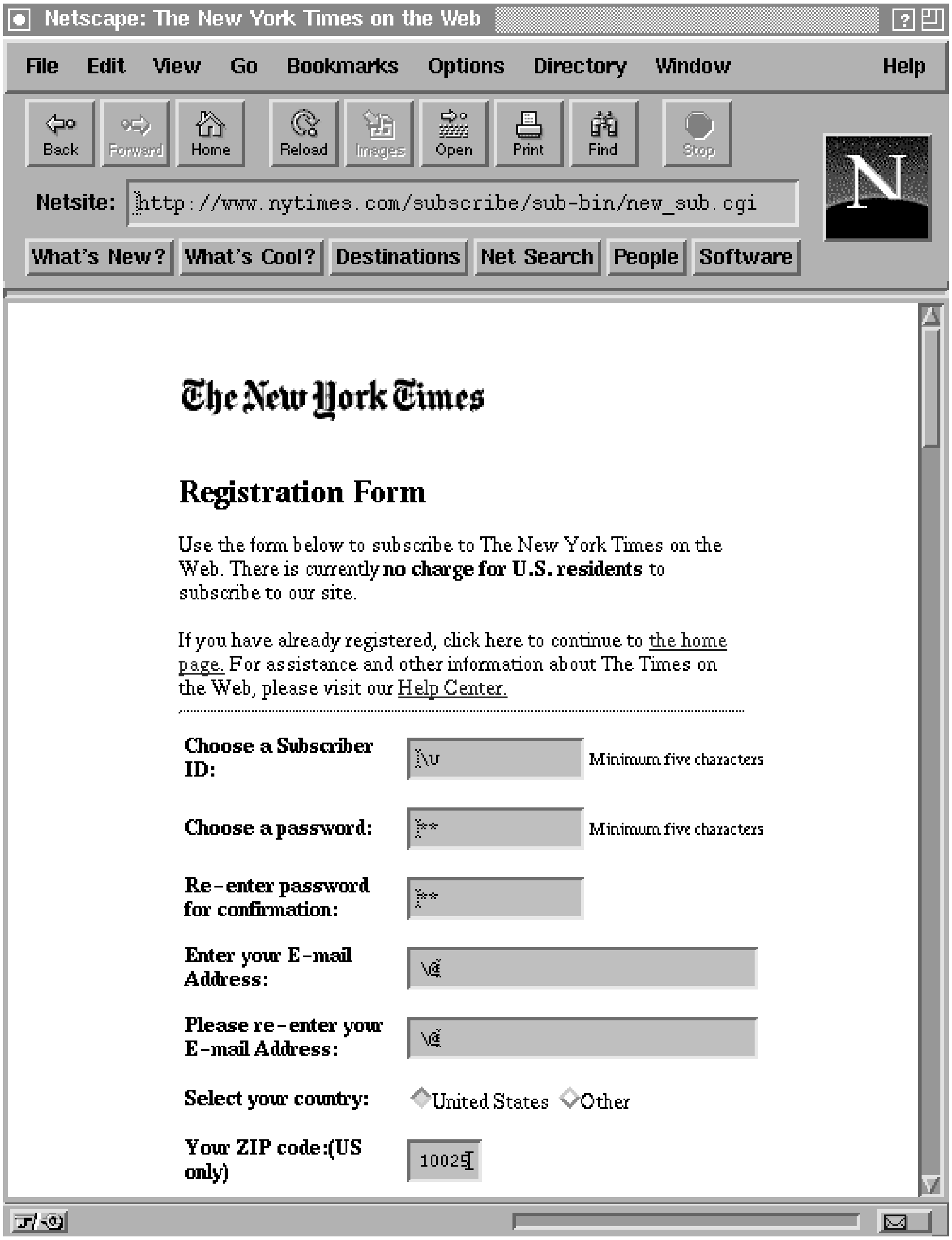

Figure 3: New York Times registration page ©1997

meant that we could easily stop, restart, or replace the server. The system could easily recover from server or machine crashes. Furthermore, if there were a wide selection of LPWA proxies available worldwide, a user could use any one of them equally well. From a security standpoint, not keeping state information on the server reduces the threat to privacy. If the server had to keep user identities, an intruder could possibly obtain the identity information and learn who is using the server.

Statelessness would be less important if the proxy server were to reside on an intranet's firewall, as described in [BGGMM98]. In that case, the server is within a trusted environment, and keeping state allows for some interesting extensions (see [BGGMM98]).

Follow the HTTP Standard: Adhering to published standards renders LPWA more widely usable, which was our goal.

3.2 Management of User ID and Secret

Given the above requirements regarding statelessness, we needed to find a way to coax a web browser to remember the user-supplied information, and to forward it to the LPWA proxy with each HTTP request. An obvious choice is to use the HTTP's `Proxy-Authorization` header. Before further discussions, we need a little review of HTTP.

3.2.1 A Little HTTP Review

The Hypertext Transfer Protocol (HTTP) is the controlling protocol for the WWW. HTTP is a stateless protocol between a client and a server. A client (typically a web browser) connects to a server, sends a *request line* and zero or more *request headers*, and, possibly, a *message body* (such as the contents of a form), and awaits a response. The server responds with a *status line*, zero or more *response headers*, and, usually, a message body (such as a web page). After this transaction, both sides will close the connection. (More recent versions of HTTP allow for both sides to keep the connection open, but any subsequent requests are treated as logically independent.) An *HTTP proxy server* acts as a go-between, sitting between the user's web browser and the intended server (denoted the *origin server*). To the user's browser, the proxy behaves like a server; to the origin server it behaves like a client. When a browser connects to the proxy, the proxy must interpret the request and make its own request to the origin server. It must then interpret the response from the origin server and pass the response along to the browser.

3.3 How to Keep State in a Stateless Proxy

One of our design requirements was that the LPWA HTTP proxy should be stateless, which meant that the proxy could not retain the User ID and Secret for active browsing sessions from one request to the next. However, for LPWA to be as easy to use as possible, the user should have to enter her User ID and Secret at most once per session. We resolved this contradiction by inducing the browser to tag each user request with the User ID and Secret information. The LPWA proxy uses this information whenever it has to compute an alias. In all other cases this information is discarded. The next section describes the mechanism we used to tag HTTP requests.

3.3.1 Using Proxy Authentication

In HTTP, a proxy may require *user authentication*. Typically authentication is required to verify that a user is authorized to use the proxy. LPWA uses the mechanism for another purpose. A proxy demands authentication by answering an HTTP request with a response that contains an appropriate (error) status code and a response header. Upon seeing the particular status code and header, a browser presents a dialog box to the user that asks for a Username and Password for the proxy. After the user fills in the information, the browser repeats the original request, this time adding a `Proxy-Authorization` request header with the request; it contains the Username and Password. Thereafter, every request that the browser sends to the proxy includes the same `Proxy-Authorization` request header. A normal proxy would verify that the information in `Proxy-Authorization` matched some table of authorized users, but LPWA uses it differently. The fact that the `Proxy-Authorization` request header accompanies every request was exactly the kind of mechanism we needed for LPWA. The proxy authentication Username and Password serve as the LPWA User ID and Secret. LPWA removes the `Proxy-Authorization` request header before it forwards the request.

3.3.2 The LPWA Login Process

Our design of the LPWA login sequence went through four iterations. We modified the Apache proxy code so it would only forward requests that included a well-formed `Proxy-Authorization` header. Otherwise the proxy rejected the request, as outlined above, which induced the browser to ask the user for authentication information (User ID and Secret).

Version 1: Proxy Authentication Only. The first version of the LPWA login sequence simply used the modified proxy, as just described. We quickly discovered, however, that this approach was inadequate. The user had only one chance to get the information in the proxy authentication dialog right, by which we mean consistent with previous uses. There was no easy way to detect typos. User typos would cause LPWA to create aliases that differ from the user's previous sessions with LPWA, and hence cause all the user's attempts to login to previously visited web-sites to fail.

Version 2: Verification. In the second iteration, LPWA presents a form to the user, asking for User ID and Secret. After the user completes this form, LPWA executes the proxy authentication step as in Version 1 and then compares the UserID and Secret supplied in the form against the corresponding `Proxy-Authorization` header information.

To implement this approach, the proxy code was changed as follows: If the user's request contains no `Proxy-Authorization` header, the proxy sends a special form whose behavior is closely coupled with the proxy's. The form processing is implemented using a Common Gateway Interface (CGI) script on the proxy, rather than by building the processing into the server code. The form asks the user for a User ID (e-mail address) and a Secret. The form-processing script on the proxy verifies (1) that both fields are complete; (2) that the User ID parses like an e-mail address; (3) that the secret is at least six characters long; and (4) finally checks whether there is a `Proxy-Authorization` header. If not, it demands one from the browser as in Version 1. Once it gets the header, the script checks that (1) both User ID and Password are supplied in the header and that the form information agrees with them. If all the checks succeed, the script sends a redirection response to the browser, giving it the URL of the user's original request (typically, the user's start-up page). This has the effect of causing the browser to try that URL again. Now that each request contains a `Proxy-Authorization` header, and now that the proxy (and script) has made sure it is correct, the proxy does not intervene, and the request gets processed normally. After the LPWA login sequence succeeds, the browser immediately goes to the originally requested web page.

Version 3: Verification and Notification. After Version 2 had been in use for awhile, we realized there was something missing. We wanted to inform the user that the LPWA login had succeeded and to remind her that LPWA would not ask them again for identity information (to prevent a rogue web-site from trying to spoof the LPWA login sequence and grab her User ID and Secret). The change from the previous version of the login sequence to this version was slight. We changed the proxy's login script to display a "confirmation" page after the login process succeeded. This page informs the user that her LPWA login succeeded. It also serves as a bulletin board for LPWA announcements. A button on the "confirmation" page takes the user to the originally requested web page.

Version 4: Enter-Once Login The change from Version 1 to Version 2 required the user to type the User ID and Secret twice, once into the form and a second time into the proxy authentication window. Feedback from LPWA users indicated that this was cumbersome. We decided to give users a choice. They could choose the "safer" login, which required the double entry of information, or use a login more like Version 1. To better support Version 1, we decided to store a cryptographic hash value of each UserID/Secret pair encountered by the proxy, so the proxy could distinguish between first time and repeat users (without inferring their identities), and provide distinct greetings. Thus when a repeat user mistypes her UserID and Secret, the proxy greets her as a first time user, which alerts her to the mistake. The greeting page provides the user a second chance to login, so that any user so alerted can correct her mistake. This addition is a slight departure from the statelessness requirement, but was well received by LPWA users.

3.4 The LPWA Proxy in Use

After successfully logging into LPWA, a user surfs the Web transparently with respect to LPWA. But note that each HTTP request by the user and each answer by the web-site is routed through LPWA, thus providing the required *indirection*. A user explicitly invokes LPWA, whenever a persona is needed, by typing one of the following LPWA escapes:

Escape	Used for...
`\U`	(alias) username, nickname, first name, etc.
`\P`	(alias) password
`\@`	(alias) e-mail address

Users can supply these escapes in two contexts: in HTML forms, and as identity information for HTTP *basic authentication*. Basic authentication is similar to the previously described proxy authentication, except that the origin server, instead of the LPWA proxy, demands the authentication information.

Forms. An HTML form that contains user-supplied data can get sent to an origin server in two ways: either as the message body of an HTTP POST request, or as the *query-string* of an HTTP GET request. In the case of the POST request, the message body contains (attribute, value) pairs. LPWA parses the message body and inspects each value to see whether it matches one of the LPWA escapes. If it matches, LPWA substitutes the corresponding persona information. The persona is computed via the *Janus function* [BGGMM98], which takes as input the User ID, Secret and web-site domain name. After LPWA checks (and substitutes for) all (attribute, value) pairs, it repackages the message body so it can be sent to the origin server. This includes recalculating the `Content-Length` request header, which indicates the size of the message body. In the case of the GET request, LPWA examines the query-string, which is part of the request URL. The processing is similar to the POST case, in that there are (attribute, value) pairs, but the details of the parsing and reassembly are different.

Basic Authentication. An HTTP origin server can require a user to authenticate her/himself to the server. The mechanism by which this gets done is exactly analogous to the proxy authentication described above, except that the client (browser) sends an `Authorization` header that contains the username and password for the user at the server. LPWA examines the `Authorization` header to see whether the username and/or password is an LPWA escape, and, if so, it makes the appropriate substitution of persona information.

3.5 Other Proxy Processing

In the interests of enhanced privacy and security, LPWA filters HTTP request headers. Specifically,

- The `From` header, which is seldom used, but which could contain the user's real e-mail address, is removed.
- The `User-Agent` header, which can disclose information about what type of machine the user has, is trimmed to remove the platform-specific information. The latter is a potential hint for a hacker trying to break into the user's machine.
- The `Referer` [*sic*] header is removed. `Referer` contains the URL of the web page in which the URL of the current request appeared. Thus it permits a server to learn the previous page the user visited, which may contain personal information, especially if it is a user's home page, "personal favorites" page, or information about the user's organization. The problem with removing this header is that there are sites which restrict access based on the value of the `Referer` field. For example, one web-site of syndicated comic-strips restricts access to requests where the `Referer` has the value of a newspaper site. We accommodate such cases via a configuration file. This is discussed further in Section 5.2.

4 Design of LPWA E-mail Forwarding

As described earlier, the LPWA proxy creates an alias e-mail address for users in response to the `\@` escape in forms. In [BGGMM98], we describe an e-mail scheme in which the alias e-mail address generated is the alias username at an appropriate domain; `lpwa.com` in our case. The e-mail system then stores incoming messages, and a user agent retrieves messages for all aliases that belong to a particular user. This scheme has the advantage that the alias e-mail address generation is trivial and that no privacy-compromising information has to be stored on the e-mail system. However, such a scheme is better suited for environments in which the proxy resides on a firewall or an ISP access point.

In our trial configuration as an external proxy, a user typically expects e-mail to be forwarded to her real mailbox. In [GGMM97], we describe such a scheme and show that the resulting alias e-mail address has the same desirable properties as the alias username and password. Actively forwarding without maintaining state implies that the alias e-mail address is some sort of *encryption* of the user's real e-mail address (User ID). The drawback of such a scheme is that the proxy and the forwarder must store the secret encryption/decryption key. Possession of this key compromises user privacy, and hence security of this key is paramount. Note that storing the encryption/decryption key does not contradict the statelessness of the proxy, since the key is fixed and may be considered as a part of the proxy code.

While implementing LPWA, we quickly noticed that many web-sites limit e-mail addresses in registration forms to some arbitrary and rather small length. The method of [GGMM97] produced alias e-mail addresses that were too long. Hence, we had to resort to a more heuristic approach: We first compress the user's real e-mail address, which is the LPWA User ID, and then encrypt it to generate the *mailbox* part of the alias address. See Appendix A for more details. The *domain name* part of the alias address is the address of the machine that runs the LPWA e-mail forwarding software. The forwarding software is derived from a Simple Mail Transport Protocol (SMTP) gateway daemon that was written at Bell Labs. It was modified so that the incoming mailbox name is decrypted to reverse the previous encryption. If the decryption fails to result in a valid e-mail address (according to RFC 822 [RFC822]), the forwarder rejects the e-mail, and it writes a log entry. Otherwise the forwarder uses the host system's e-mail subsystem (*sendmail*, in our case) to forward the e-mail on to the true recipient.

4.1 Anti-Spam Tool

As part of the Persona Generator, a user obtains a different and seemingly unrelated alias e-mail address for each web-site for which she registered. For example, a user might be known as `hwfyh8yocY8XUKm9t5OKvnNW@lpwa.com` to `my.yahoo.com` and as `lN8illidPtFk5OSthNoXzGuS@lpwa.com` to `www.expedia.com` . This feature enables effective filtering of junk e-mail (spam), as follows.

Whenever the LPWA E-mail Forwarder decrypts an alias e-mail address in order to forward a message to the user's real e-mail address, it includes the alias e-mail address in the `CC` e-mail header of the forwarded message. We decided to use the `CC` field, since many commercial e-mail readers support filtering of incoming e-mail messages based on this field.

Assume that a user registers at `www.crook.com` and LPWA gives `bd1YnEWOmot3CX-_UxonbznP@lpwa.com` as the alias e-mail address. Now the address database at `crook.com` gets sold to spammers. As soon as the user gets the first piece of junk e-mail, she can install a local mail filter for the string `bd1YnEWOmot3CX-_UxonbznP`. This will eliminate all e-mail caused by the selling of the crook's database to spammers, while at the same time e-mail from all other sites is unaffected. Most current anti-spam tools filter according to sender addresses or keywords, both of which are easily changed by spammers (e.g., address spoofing). Our method is the first to filter according to the *recipient* address. A spammer who bought the address database from crook knows the user only as `bd1YnEWOmot3CX-_UxonbznP@lpwa.com` and hence cannot change (spoof) this string!

Furthermore, the user can easily keep a small local database, mapping alias e-mail addresses to the web-site for which the address was created. Then, when receiving junk e-mail, the user can determine which web-site is responsible, even when the junk e-mail was sent by a third party. The user can complain to the web-site or take other action, as needed.

5 Experiences During Testing and Trial

After extensive internal testing, we announced a public LPWA trial that became operational in June, 1997. In this section, we describe our experience during both the testing and the (ongoing) public trial.

As described in the previous sections, the LPWA design is conceptually simple. But the devil (or God) is, as always, in the details. Initially we solved the problems we encountered by building the changes directly into the LPWA proxy source. Eventually it became evident that what we needed was to add *directives* to our modified Apache proxy server, so we could change things easily through the server's configuration file. We will describe the added directives as we go along.

5.1 Logging In Failed After Registration

The first problem we encountered was that we could register at some sites, but when we tried to log in there subsequently, the login failed. We traced the failure to the fact that, at some sites, the domain name for the machine that handles registration is different from the one that handles return visits. For example, the New York Times site is `www.nytimes.com`, but registration is at `verify.nytimes.com`. So a user who registers there would get a site-specific identity for the `veryify.nytimes.com`. But when the user returns to the `www.nytimes.com` and enters the LPWA escapes in the login form, LPWA produces an identity that fails to match the one it produced during registration, and hence the login fails. We decided that this behavior would likely occur often enough that we should treat all domain names of the form `site.company.com` as if the domain name were `company.com` for the purpose of generating LPWA aliases. Our generalization was to add a directive that instructs the proxy to look for a suffix and retain a specified number of parts of the domain name. In this case the suffix is `.com`, and the number of components to retain is two.

A second variant of the same problem occurred when a web-site used the IP address of a particular machine in a URL, rather than its name, to handle registration. Because we wanted that IP address to behave identically to the name the web-site usually uses, we added a directive to the proxy that asserts the equivalence of one name (or IP address) with another. A related variant arose when we found web-sites that share identity information. For example, users can also use their identity from `www.eonline.com` at `www.moviefinder.com`. We were able to use the same equivalence directive as above to assert the equivalence of the latter site to the former. Thus a user's identity at `www.moviefinder.com` is identical to her identity at `www.eonline.com`.

5.2 Services That Failed

We discovered that services at some web-sites simply stopped working when LPWA was interposed. We traced the problem to the fact that (at least initially) LPWA unconditionally removed the `Referer` header to improve privacy protection. We re-evaluated LPWA's behavior on a case-by-case basis. First, we discovered that some of those failing sites merely insisted on seeing a `Referer` header, without actually inspecting its value. Initially we obliged by always sending `Referer` with a null value.

But other sites were more demanding. The `www.uclick.com` site, for example, distributes comic strips on the web, and it acts as a service for subscribing newspapers. It therefore requires that the `Referer` field contain the domain name of one of its subscribers. We obliged by always passing `Referer` to `www.uclick.com`, reasoning that the `Referer` field must be from a subscribing newspaper and was therefore unlikely to disclose information about the user. We discovered that two other sites failed to work correctly unless the `Referer` field named its own site: `www.tvguide.com` and `www.wired.com`. Using the same reasoning as above, we pass `Referer` for those sites, too.

We addressed this issue by adding a server directive that allowed us to selectively pass `Referer`, based on the origin server to which the request is being sent and the name of the server in the `Referer` header. For example, we allow `Referer` to be sent to `www.uclick.com` from any (`Referer`) URL. However, we only forward `Referer` to `www.tvguide.com` if the `Referer` is also `www.tvguide.com`.

5.3 E-mail That Is Not Delivered

As discussed in Section 4, one of the problems with our approach to handling e-mail is that the persona e-mail addresses are typically longer than the original addresses. The *mailbox name* part of the persona address is about the same length as the true e-mail address. Adding the *domain name* part corresponding to the forwarding machine often produced an alias that no longer fit in the space provided by some registration forms. Like the login process, our handling of e-mail went through several generations, as described below.

Canonical E-mail Addresses. The first response to the length problem was to develop increasingly more compact encodings/encryptions of the mailbox name. These encodings had to conform to the relevant Internet standards (RFC 821 [RFC821]), yet remain invertible. To increase the coding alphabet, we used both upper and lower case characters for the encodings. To our surprise (too naive?), we discovered that many web-sites force e-mail addresses supplied in forms to all lower case. While doing so for the domain name is acceptable, RFC 821 specifically states that mailbox names are **case-sensitive**. Our response was two-fold. First, we sent e-mail to sites where we saw this behavior, noting that they were possibly reducing their user community for those people whose mailbox names were, indeed, case-sensitive. Second, we devised another encoding that was case insensitive. This encoding included a (non-alphanumeric) flag character to identify it. (Unfortunately, since this encoding has a smaller coding alphabet, the resulting addresses are longer.) Then we added a directive to the server to specify that the generated e-mail address for specifically named sites should use the lower-case alternative encoding. We updated the proxy's configuration as we encountered such sites, but of course that meant the first (few) registrants using LPWA would not get their e-mail delivered, because, when they registered, LPWA generated mixed-case e-mail addresses for the site. We subsequently found sites that forced e-mail addresses to all **upper** case. We therefore had to change the mailbox decoding function so that, rather than handle lower-case-only encodings, it would be case-insensitive, with the caveat that the encoding had to be all one case.

Call Off The Police. We discovered that some popular web-sites that mishandled mixed-case mailboxes *also* rejected during their registration process the LPWA e-mail address with our chosen flag character (specifically, ' `$` '), but those sites would accept e-mail mailboxes that contain a different flag character, ' `/` '. So we changed the default flag character.

Although we (naively) assumed that most web-sites would handle e-mail addresses correctly, the painful evidence was that, in fact, few did. The job of configuring the LPWA proxy to generate different encodings depending on web-site, and of notifying web-sites of their violations of the Internet e-mail standard had become onerous. So we decided to capitulate to reality and to generate the same mono-case e-mail encoding (with the ' `/` ' flag character) for all sites.

Our Final Capitulation. We soon discovered that our "solution" was still unacceptable. Some sites for which ' `$` ' was an acceptable flag character during registration did not accept ' `/` '. We had to move to yet one more e-mail encoding to use everywhere. The final encoding, therefore, is an encoding comprising single-case alphabetic and numeric characters, with no flag characters. Meanwhile, the SMTP server accepts all the previous encodings, along with the new one, so e-mail from the sites that did handle e-mail correctly continued to be delivered successfully.

6 Related Work

Our work provides *data anonymity*, which protects the identity of the user by careful modification of the data she exchanges with the world. The other type of anonymity is *connection anonymity*, which protects the identity of the user by disguising the communication path between her and the rest of the world. LPWA provides a limited connection anonymity by using an HTTP proxy. However, tracing all communication to and from the proxy may reveal the user's identity.

The *Anonymizer* (see [Anon]) is a service which provides limited data and connection anonymity. It is an intermediate entity which filters HTTP headers for web browsing, and rewrites all HTTP pages so that clicking on one of the links causes a request to be sent to the Anonymizer server, which in turn issues the original request. However, there are no features provided for anonymous registration at web-sites, and hence no simple and secure means for users to preserve data anonymity at web-sites that offer personalized services.

Onion Routing [SGR97] and *Crowds* [RR97] are two recent systems that provide a high degree of connection anonymity for Web browsing. Similar to mixmaster remailers, Onion Routing transforms a message into several layers of encryptions ("onions"). Each layer determines the next forwarding node ("onion router"). To enable two-way communication, onion routers maintain connection state. Crowds randomly assigns a native route for each crowd's member ("jondo") among other jondo's before the connection is routed outside the crowd. We note that LPWA can be potentially combined with these tools to give a high degree of both data and connection anonymity.

See [M98] for a recent overview of Internet anonymizing techniques.

The companion paper [BGGMM98] contains additional references to the theoretical aspects of alias generation.

7 Performance

Introducing a proxy between the user's browser and the origin server will always produce a performance penalty because of the extra-hop TCP/IP connection. We wanted to verify that LPWA's processing was otherwise inconsequential, and indeed it was. The actual proxy processing delay, that is, the time between when the proxy read an HTTP request from the browser and started to make a new request to the origin server (and not counting the extra TCP connection), was about 4 ms. (on a 166MHz Pentium, running Linux). The total CPU time that the proxy required to process a request and the corresponding reply was about 10 ms. The time to process requests that contained LPWA escape sequences was unmeasurably different from requests without them. By contrast, the time to set up the connection from the client to the proxy (the client and proxy were "close") was about 35 ms. In other words, the LPWA proxy running on this hardware could handle more than 50 requests per second with 50% CPU utilization.

Clearly the best way to minimize the performance impact of an LPWA HTTP proxy is to place it as close to users as possible. In a dial-up ISP setting, that would mean putting the proxy close to the dial-up access servers. In a corporate setting, that would mean putting the proxy near the corporate firewall.

8 Conclusions

The LPWA trial has run since June, 1997, and has thus far attracted over 21000 unique users (by May, 1998). About 40% of those users have logged in more than once. For the last few months, an average of 400 to 500 distinct returning users log into LPWA every day. In order to count the users without compromising their anonymity, the LPWA proxy logs the one-way hash value of the User ID and Secret.

The trial version of LPWA is currently available at `lpwa.com` . Apart from network problems early on and some later hardware failures, the LPWA proxy has run smoothly. Likewise, the e-mail forwarding software has run well (for correctly supplied To addresses), forwarding several hundred messages per day.

Based on the number of users logging in, and on the network traffic, we believe that the ongoing trial has been a success. The LPWA user base has grown steadily, despite performance degradation for those users whose location is "inconvenient" with respect to the proxy's location in Murray Hill, New Jersey. LPWA has also won recognition in the trade press, featured in InternetWeek, Inter@ctive Week, Wired Magazine, The New Your Times' Cybertimes, and C—Net News, and selected as PC Magazine's developer's site of the week, Business Week's innovation of the week, and Wired Magazine's "Just Outta Beta".

A Generation of an Alias E-mail Address

In this appendix, we describe in further detail the algorithm used for generating a working alias e-mail address from a user's real e-mail address (User ID) and a web-site domain name. As explained in Section 4, we could neither use the passive mailbox approach of [BGGMM98] (we need active forwarding) nor the direct e-mail forwarding approach of [GGMM97] (we need shorter e-mail addresses). We settled for the following heuristic approach, which seems to approximate the desired properties (see [BGGMM98]) of an alias e-mail address reasonably well. LPWA creates alias e-mail addresses that are effectively unique for a given User ID and web-site (there is some negligible probability that a user's alias e-mail addresses at two distinct web-sites will be the same). The algorithm uses a secret key that is known to both the LPWA Persona Generator (*i.e.*, the LPWA proxy) and the LPWA E-mail Forwarder.

A.1 Compression

The user's real e-mail address is subjected to a variable-length compression. The goal is to remove redundant bits in the original by turning the character-based e-mail address into a more compact binary representation through the following series of steps.

1. The user's e-mail address is split into its *mailbox* and *domain name* components.
2. Both the mailbox and the domain name are checked to see whether all their characters are in the following 31-character alphabet (called the *5-bit encoding*): `a-z` , `0` , `1` , `-` , `.` , and `@` . So, for example, both the mailbox and the domain name of `dmk@bell-labs.com` can be 5-bit encoded, but only the mailbox of `dmk@3com.com` can be 5-bit encoded since the character `3` in the domain name is not in the 5-bit alphabet.
3. The top-level domain (TLD), the last part of the domain name (*i.e.*, `.com` in the above examples), is looked up in a table of TLDs. Some TLDs have an extra-short encoding, some have a short encoding, and some have no special encoding.
4. If neither the mailbox nor the domain name can be 5-bit encoded and if there is no TLD encoding, the e-mail address is simply 7-bit encoded and compression is complete. In 7-bit encoding, the low-order seven bits of consecutive characters are compressed into 8-bit bytes.

5. Otherwise, the first byte of the encoding contains flags that describe whether the e-mail address has 5-bit encoding (both the mailbox and the domain name can be 5-bit encoded), 5-7-bit encoding (the mailbox can be 5-bit encoded, but the domain name requires 7-bit encoding), or 7-bit encoded. Orthogonally, other flags describe the TLD encoding: either extra-short (a few bits), or short, in which case the TLD number is stored in the next byte.
6. Following the flag byte(s), the mailbox and then the domain name are packed together using five or seven bits per character, depending on whether each can be 5- or 7-bit encoded.
7. The result is padded at its end with a variable number (1-4) of `NUL` characters, the number depending on a byte in the MD5 hash of the destination web-site domain name. (The padding ensures that a given user's alias e-mail addresses vary in length; this protects against web-sites using e-mail address length to learn that e-mail addresses at distinct web-sites belong to the same user.)

A.2 Encryption and Encoding

A single byte B from the MD5 hash of the web-site domain name (a different byte from the one used in compression) is concatenated to a fixed symmetric key, thus making the encryption key site-dependent. The compressed e-mail address is encrypted with this key using CBC-DES, and B is then concatenated to the result. If the resulting e-mail address is to be mono-case, the result of the concatenation is encoded using the characters `a-z` and `0-5`. For mixed-case e-mail addresses, the result of the concatenation is encoded using the characters `A-Z`, `a-z`, `0-9`, `_`, and `-`. For example,

`hwfyh8yocY8XUKm9t5OKvnNW`

is a valid mixed-case mailbox encoding, and

`cupupg3faxer3lmzhl3t4fqdjsl45`

is a valid mono-case encoding for the e-mail address `lpwa@research.bell-labs.com`.

Note that the combination of four possible NUL byte paddings plus 256 possible encryption keys (because of byte B) leads to 1024 possible LPWA e-mail aliases for a given user's e-mail address.

References

[Anon] THE ANONYMIZER. `http://www.anonymizer.com`.

[BGGMM98] D. BLEICHENBACHER, E. GABBER, P.B. GIBBONS, Y. MATIAS, A. MAYER, On secure and pseudonymous client-relationships with multiple servers. To appear in *Proc. 3rd USENIX Electronic Commerce Workshop*, August 1998.

[GGMM97] E. GABBER, P.B. GIBBONS, Y. MATIAS, A. MAYER, How to make personalized web browsing simple, secure, and anonymous. In *Proc. of Financial Cryptography'97*, Springer-Verlag LNCS 1318.

[M98] D. MARTIN, Internet anonymizing techniques. *;login: Magazine*(The USENIX Association Magazine), May 1998, pp. 34–39.

[RFC821] J.B. Postel, Simple mail transport protocol. Available at ftp://ftp.isi.edu/in-notes/rfc821.txt .

[RFC822] D.H. Crocker, Standard for the format of ARPA internet text messages. Available at ftp://ftp.isi.edu/in-notes/rfc822.txt .

[RFC2068] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, Hypertext transfer protocol — HTTP/1.1. Available at ftp://ftp.isi.edu/in-notes/rfc2068.txt .

[RR97] M. Reiter, A. Rubin, Crowds: Anonymous web transactions. *ACM Transactions on Information and System Security*, to appear. Manuscript at http://www.research.att.com/projects/crowds/ .

[SGR97] P. Syverson, D. Goldschlag, M. Reed, Anonymous connections and onion routing. In *Proc. IEEE Symposium on Security and Privacy*, 1997.