A Security Architecture for the Post-PC World

Steve Ross, Jason Hill, Mike Y. Chen, Anthony D. Joseph, David E. Culler, and Eric A. Brewer
Computer Science Division
University of California, Berkeley
{stevross,jhill,mikechen,brewer,culler,adj}@cs.berkeley.edu

Abstract

The Internet has made a large variety of services and content available to users, including secure access to electronic commerce and personal information. Simultaneously, the Post-PC revolution is bringing information access to a wide-range of devices beyond the desktop, such as public kiosks, and mobile devices like cellular telephones and PDAs. We propose the use of an infrastructure-based security proxy architecture to bridge the gap between the capabilities of Post-PC devices and the security requirements of Internet services. By combining generic content and security transformation functions with service-specific rules, the architecture decouples device capabilities from service requirements and simplifies the addition of new devices and services. The architecture offers the novel ability to deal with untrusted public Internet access points by providing fine-grain control over the content and functionality exposed to the end device, as well as support for using trusted and untrusted devices in tandem.

1 Introduction

Internet services for electronic communication and commerce have permeated our society. People routinely access and exchange private and sensitive information when they use Internet services, such as electronic mail, the World-Wide Web, electronic banking, on-line shopping, E-commerce, and stock trading. At the same time, we are moving into a "Post-PC" era, where people use many different types of devices to access these Internet services, including pay-per-use public Internet kiosks, laptops, PDAs, cellular telephones, and two-way pagers, each with different characteristics and capabilities. Three critical requirements for users of Internet services in these Post-PC environments are *secure access to information, dynamic content adaptation,* and the *fusion of multiple devices*. The *fusion of multiple devices* represents an extension of the split trust model [5] where multiple devices work in tandem. Meeting these requirements will preserve the privacy and integrity of users' transactions and enable the use of new and interesting services and access devices.

Secure, wide-area access to information is of tantamount importance on the Internet, but existing solutions do not address Post-PC requirements. It is recognized that traffic is readily intercepted and user identities can be forged. Protection against such attacks is provided by strong authentication and end-to-end encryption. A user's *trusted* application establishes a secure connection to the service of interest. For example, when accessing a secure web-based banking service, the user's web browser establishes a secure connection to the web server by using HTTP over Secure Socket Layer (SSL) [17].

The existing security model makes two basic assumptions: first, that both the user's access device and the software running on it can be trusted not to intercept or send private information elsewhere; second, the access device has the computational resources to secure the connection (e.g., to perform public key SSL operations) and to display the service's content (e.g., to render complex color graphics). While these assumptions are reasonable for users of private desktop computers, they do not hold for Post-PC devices, where users access services from both untrusted devices (such as public kiosks), and devices with limited resources (such as PDAs).

Untrusted kiosks are problematic for secure Internet services, as all service data is available in unencrypted form to the kiosk. Possible attacks may be as straightforward as recording all keystrokes (e.g., Personal Identification Numbers, passwords, and login information), or as subtle as recording users' personal information for later fraudulent use (e.g., account numbers and home addresses). Additionally, a kiosk can "hijack" a secure connection and perform active attacks (e.g., make bank transfers or send forged e-mails). Untrusted endpoints should not be allowed to see a user's personal information; *instead, the content value of such data must be re-*

duced. These endpoints also should not be able to access sensitive service functionality; instead, the service functionality accessible must be guarded.

PDAs are also problematic because they are generally low-power, computationally-limited devices with limited memory and networking capabilities. To perform the industry-standard SSL handshake phase on one such device, a 3Com Palm Pilot V [26] requires several seconds. This latency imposes an intolerable delay for connection setup, which is particularly undesirable if network connectivity is intermittent. An SSL implementation that uses elliptic curve cryptography [7] is feasible on a Palm Pilot V, but few Internet services support that option. Moreover, even if a PDA is capable of performing SSL, one may still opt for a protocol that is faster and more power-efficient. Finally, entering data to fill out forms using a pen-based interface is tedious at best and even more cumbersome using number pads on devices such as cellular telephones.

We propose the use of a trusted infrastructure-based proxy service to solve these problems. For untrusted terminals, sensitive user information is removed from the information stream going to the terminal. For example, real names and account numbers can be screened out. The proxy also mediates access to the service by filtering control data in order to limit allowable actions (e.g., disallowing all stock trades on untrusted kiosks). For access from computationally-limited devices, the proxy service transcodes the security protocol into one which is more efficiently executed on the device. Additionally, the proxy can distill the service content into a format more suitable for the device [15].

A drawback of the proxy approach is that it requires users to place a significant amount of trust in the proxy infrastructure. The components in the infrastructure require access to all data flowing through them to perform filtering and adaptation transformations. Additionally, users are required to store private information and preferences in the infrastructure to enable these operations. This trust can be mediated by the fact that the security proxy itself can be co-located with a particular Internet service, in which the user already has established trust of sensitive information. For example, an Internet e-mail or stock-trading service can provide its own security proxy for access from mobile devices and public terminals. Alternately, security proxies can be treated as orthogonal to the services which they enable access to; a user need only establish an account with a security proxy provider, much as they do with an Internet service provider today.

In addition to enabling basic access, our security proxy can be used to combine the capabilities of both untrusted public terminals and mobile personal devices. For example, the limited GUI of a PDA can be supplemented by the richer, larger display of a public kiosk, while the PDA is used only for sensitive operations. The security proxy splits trust between the PDA and the public terminal by fusing the devices together to provide one logical channel with secure access to the end service. Consider the case of users accessing their stock trading accounts from public access terminals. Instead of relying on the terminal to protect their secure information, the users could direct private or sensitive information (e.g., portfolio value and account information) to their portable device, while using the GUI capabilities of the public terminal to initiate requests and display generic stock information (e.g., stock price fluctuations and historical graphs). The users initiate trading operations through the untrusted public terminals and then confirm them using their trusted portable devices. The connections to the mobile devices can be provided by the environment (e.g., kiosks with infrared network connections for PDAs) or by the devices (e.g., receiving a call on a cell-phone, or a message on a two-way pager).

In this paper, we present a secure proxy architecture that:

- Allows users to protect themselves from untrusted points of access.
- Allows users to use untrusted points of access in tandem with trusted mobile devices for security-enhanced service interactions.
- Provides users with the flexibility to use their access devices of choice regardless of their computational abilities.

The security proxy we present here incorporates a component-oriented, rule-based architecture which minimizes the amount of service-specific code that must be written. In particular, service developers (and users) can specify security rules that make the appropriate modifications to the data and control actions flowing in both directions between the user and the service.

The design we propose is a generalization of our experience with a prototype implementation. The prototype implementation demonstrates secure access to a stock trading service from untrusted public Internet access points and resource-poor PDAs.

2 Architecture

In this section, we present an overview of our infrastructure-based proxy architecture, followed by a high-level walk through of an example scenario. We then discuss the design of each architectural component in detail.

2.1 Design Overview

Our security proxy provides a secure level of indirection that can be used to modify content flowing between clients and services. It is a component-based system for creating customizable building blocks that control and modify users' connections to secure services. These components can easily be composed together to bridge the gap between traditional services and Post-PC devices that have different trust levels and to simplify the process of adding support for new services and devices. This model provides a partitioning of computation that allows the components to be placed in a scalable, fault-tolerant, and highly-available execution environment such as [3] that meets the critical requirements of Internet-scale services.

Our architecture allows the user to take on different *trust profiles* based upon the location, device, capabilities, or trust in an access device. For example, these profiles include connecting from home devices (fully trusted), work computers (partially trusted), public terminals (not trusted), PDAs (trusted, but not powerful), or with a untrusted device in tandem with a trusted one (split-trust).

By dividing the functionality of our architecture into structural and semantic transformations, it is relatively easy to add new services and devices. *Format transformations* are used for adapting to device capabilities (e.g., changing the resolution, color depth, or encoding format of an image), while *semantic transformations* are used to protect users' data and prevent various actions from being performed from untrusted end devices. Additionally, the use of rule-based components to add service-specific semantics to generic transformations simplifies the incorporation of new services.

The components of our architecture fall into the following categories:

- Security Adaptors (SA): These components bridge the gap between the secure access protocol required by a service (server-side) and access devices (client-side).

- Format Transcoders (FT): These components perform structural transformations to convert the format of data flowing to and from services or devices into a representation that the Filter and Control Modifier component can operate upon.

- Filter and Control Modifier (FCM): The FCM is a rule-driven engine that performs semantic transformations to reduce the content value of data and restrict control actions.

- Identity Service (IS): The IS a secure central repository that stores persistent state on behalf of users. The state includes identities for accessing services and rules for displaying and modifying service content based upon the users' role and mode of access.

- Transient Store (TS): The TS stores temporary state, such as current session identifiers, mappings for obfuscated or hidden data, Uniform Resource Locators (URLs), and client request information.

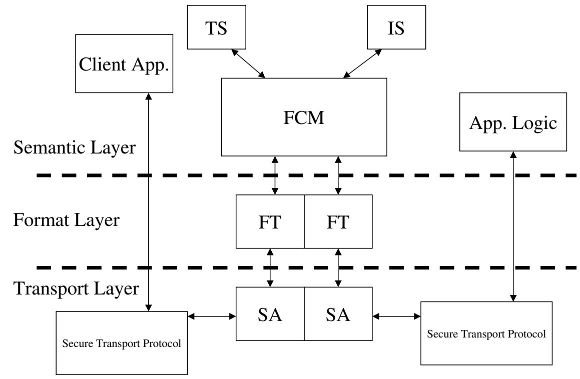

Figure 1: **The Secure Proxy Architecture**

Figure 1 illustrates the architecture of our security proxy. Each component unifies a collection of devices or protocols into one format, isolating service-specific details from the logical operations required to transform data. This abstraction capability becomes increasingly important as the number of devices and services grow. A new device or service can be enabled by simply creating a few device-specific components. This ease of composability is in stark contrast to modern vertical web-based applications that require the service author to individually tailor the service to each end device. Without this abstraction, these actions are an $N \times M$ task, where the introduction of a new device type requires changes to every service to add support for the device and vice versa (e.g., web-clipping [1]).

2.2 High-level Walk through

In this section, we explore the scenario of a user at an airport who wants to buy stock using an Internet-based stock trading service from an untrusted kiosk. The user would like to use the kiosk for filtered graphical data display (i.e., sensitive account information should not be displayed), and would like to confirm stock purchases using their trusted PDA. The kiosk provides Internet access to the PDA over an infrared connection.

We use this example as motivation for the component design decisions that are required to raise the bar high enough such that users feel comfortable with the security associated with their transactions. We choose stock trading as it most intuitively stresses each of the components in the architecture. We feel that the architecture applies equally well to other web based sites for electronic commerce, i.e. purchasing books or on-line banking. The architecture can also provide secure access to other Internet services such as directory services (LDAP), or e-mail (IMAP). In addition, we present further design decisions that support these scenarios.

At a high level, the flow and operation of the components are as follows: the user first authenticates herself to the system using incoming Security Adaptors both from the untrusted kiosk and the trusted PDA. The outgoing Security Adaptor establishes an SSL session to the stock trading service, both authenticating the service to the user and authenticating the user to the service.

The server-side Format Transcoder marks up the data from the stock trading service, converting it into a semantic format representation and a style sheet that is used for presentation (i.e., conversion of the semantic content representation into a device mark-up language, such as HTML). It also saves any temporary session state in the Transient Store. The Filter and Control Modifier (FCM) applies rules that operate on the semantic representation of the data to filter sensitive information, such as account numbers and user names, before the data can be displayed on an untrusted device.

The outgoing client-side Format Transcoders use the style sheet to re-render the semantic content as HTML for display on the kiosk as well as into a format suitable for display on the PDA.

The incoming client-side Format Transcoder converts requests flowing to a service into an XML representation that can be processed by the FCM which performs control filtering operations. For example, this could take the form of having the FCM block requests to alter personal information. The FCM also performs the final authorization of the stock purchase request through the trusted PDA.

2.3 Security Adaptors (SA)

Security Adaptors bridge the gap between different service requirements and different device capabilities for performing security protocols. This functionality allows devices capable of understanding one security protocol to access services that use a different protocol. The choice of security protocol used by a service or device to access a Security Adaptor depends on the services' or devices' capabilities. Security Adaptors have access to unprotected (unencrypted) user and service data and authenticate users, services, and devices to the infrastructure. Therefore, Security Adaptors must be trusted by end users.

When connecting from low-power devices, a device may be capable of shared-key authentication and Blowfish [27] encryption, but not capable of performing the complex computations required by SSL [17]. However, a service may only support authenticated access over SSL. In this case, the client-side Security Adaptor is a shared-key adaptor that obtains a user's key from the Identity Service and uses it to perform challenge-response authentication and to encrypt all communication with the client.

When connecting from an untrusted device that is capable of performing public key operations, users authenticate themselves to the infrastructure using a pseudonym identity that identifies the trust profile and access device, and a one-time use password mechanism such as S/KEY or SecurID [20, 21, 12]. This information does not need to be hidden from the untrusted endpoint as it can only be used once. Additionally, the proxy provides controlled access to the user's information in case the connection is hijacked. The FCM works together with the Security Adaptors, Format Transcoders, and Identity Service to perform this authentication.

Additionally, the client-side Security Adaptor can provide persistent connections, even if the service does not provide them. This functionality enables a client to establish a secure connection to the infrastructure once, while the infrastructure (potentially) establishes multiple connections to the service. If the connection setup time or round trip latency is high, this technique will yield a substantial performance improvement. Also, the infrastructure can establish multiple connections to different services to aggregate information (e.g., to perform a meta-search of multiple web sites).

2.4 Format Transcoders (FT)

Format Transcoders transform data between external service and device formats and the FCM's XML-based language. They abstract service- and device-specific information from the FCM. There are two types of Format Transcoders: server-side Format Transcoders and client-side Format Transcoders.

2.4.1 Server-Side Format Transcoders

The server-side Format Transcoder receives a service's content and transcodes it into an XML [32] representation of the semantic data and an XSL [13] style sheet capturing the presentation format. It does this by performing a "screen scraping"[1] operation. Due to the service-specific nature of the scraping operation, each site requires its own server-side Format Transcoder. However, by abstracting the service-specific information of each service into the Format Transcoders, we create a single place where all service-specific code is located and enable the FCM to be reused across multiple services. By semantically tagging data the FCM can apply rules, based on the user's current trust profile, to these tags and appropriately modify the data.

The author of the server-side Format Transcoder must identify the semantic content of displayed data and the actions that can be performed by a service. For web-based services, this is a substantial task that requires manual analysis and coding. Despite this fact, numerous Internet services, such as shopping engines [23, 29], perform this type of transcoding regularly.

When operating on data flowing towards a service, the format transformation merely transforms the XML representations of control actions into HTTP POST and GET requests. The code to perform this operation is reusable across multiple services. The semantic operations for transforming and filtering the arguments encoded in the XML representation are embedded in rules for the request and performed by the FCM.

2.4.2 Client-Side Format Transcoders

Client-side Format Transcoders perform a transformation on data flowing to and from the client into and out of the FCM's XML representation. As data travels to a service, HTTP POST and GET requests are transformed by the client-side Format Transcoder into an XML representation that can be operated on by the FCM to apply control filtering rules.

Consider, for example, the buy request that a user would execute in the stock trading scenario. The original HTTP POST request is translated into an XML representation where each form item is given an XML tag.

For data traveling to the client, the client-side Format Transcoder uses the XSL style sheet generated by the server-side transcoder to render the FCM-modified XML for the particular end device. For the stock trading example, the format for the kiosk is HTML, while the format for the PDA is a custom format.

2.5 Filter and Control Modifier (FCM)

The FCM implements the application-level logic for adapting data flowing between devices and services to the desired security parameters. The FCM applies rules associated with the user's current trust profile to perform content and control filtering on the XML representation of the semantic content. For example, the FCM rules can filter usernames and account numbers from data being sent to untrusted endpoints. Additionally, the FCM applies rules to provide split-trust control functionality using a trusted portable device to authorize and authenticate actions from an untrusted device.

The collection of content filtering rules can be used to obscure all session information from the untrusted device. This obfuscation is necessary to prevent the untrusted device from contacting the end service directly. Many services embed session identification information into data sent to the client in cookies, forms, or URLs. This session identifier is all that is required for the user to re-authenticate himself to the service. If this information were exposed to the untrusted device, the device could bypass our security proxy. The use of session obfuscation and replacement rules prevent this. The secure session information is never transferred to the untrusted client; instead, placeholder information is used which is meaningless to the end device, but can be transformed by the proxy into the original session information for the service.

Additionally, the user needs to be able to flag any operations that require authorization, and to change authorization devices on demand. For example, consider a user that forgets her cellular telephone at home, but has a wireless-enabled PDA with her. The infrastructure should recognize the availability of the alternative device (through either

[1] Extracting information from HTML representations of pages.

rule name	action	example
allow-content	Explicitly state that content is allowed to be displayed	allow-content `<AccountValue>`
hide	Replace content with obscuring text	hide `<UserName>`
obfuscate-well-known	Replace with well-known obscuring text	obfuscate-well-known `<Account-Number>` "Your Checking Account Number"
obfuscate-mapping	Replace with replacement text generated from set-of-mapping-data, and store mapping in Transient Store	obfuscate-mapping `<StockSymbol>` US-State-Abbreviations-Set
obfuscate-form	Replace all form field names with randomly generated mappings, and store mapping in Transient Store	obfuscate-form `<StockOrderCgiName>` e.g. form field before and after are: `<select name="buysell">` `<select name=34382>`
obfuscate-cookie	Replace cookies with randomly generated cookie storing mapping in Transient Store	obfuscate-cookies `<DatekOnLine>`
verify	Send content to trusted device for verification, placing in Transient Store	verify `<StockQuote>`

Figure 2: Content filtering rules used by the FCM

explicit or implicit notification) and allow it to be designated as the user's authorization endpoint.

The FCM operates by matching each of the XML tags in the intermediate format produced by the FT to the applicable rule, and executing the rule. Figures 2, 3 and 4 summarize the types of rules and their actions.

2.5.1 Using Content Filters to Obfuscate Data

Infrastructure content filters alter or remove sensitive service content before it reaches the untrusted device, effectively decreasing the data's privacy or security level.

The FCM applies the user's rules by matching each XML tag to the appropriate rule, and performing the specified action. In order for a data item on the page to be displayed, the security level at which it can be rendered must be explicitly specified by the user's trust profile. This requirement follows the principle of least privilege.

There are six types of rules associated with data value reduction: *allow-content*, *hide*, *obfuscate-well-known*, *obfuscate-mapping*, *obfuscate-form*, and *obfuscate-cookie*. The *allow-content* rule explicitly states that a content item is allowed to be rendered on the page in its current format. This rule prevents service format changes from causing secure information to be accidently released. The *hide* rule replaces sensitive information in the content with an uncorrelated number of "|" characters, within a reasonable limit of the original size of the data. The *obfuscate-well-known* rule replaces well-known sensitive information with its description rather than its value. For example, rather than displaying the user's home address "777 Main Street", the text "HOME ADDRESS" is displayed. This information is meaningful to the end user, but leaks no useful information to the kiosk.

The *obfuscate-mapping* rule produces a random table of mappings, stores the table in the Transient Store, and replaces sensitive data with an obscured name from the table. If a trusted device is being used in tandem with an untrusted one, the FCM sends the mapping table to the user's trusted device, so the user can make sense of the obfuscated data. In the stock trading example, to prevent the kiosk from obtaining the list of stocks contained in a portfolio, the FCM applies the *obfuscate-mapping* rule to the portfolio data. For example, the result could be a table that maps stock names to the names of states (e.g., "IBM" maps to "Kentucky", "AOL" to "Michigan", etc.).

A rule that is similar to the *obfuscate-mapping* rule is the *verify* rule, which leverages the trusted mobile device to verify content displayed on the untrusted endpoint. The *verify* rule can be used to make sure that the untrusted device does not modify useful public information in a malicious manner. For example, the price of an item on an on-line shopping web site is of dubious value if it can be modified by an untrusted endpoint. The *verify* rule uses the portable trusted device to display small amounts of data, thereby allowing the user to determine whether the untrusted kiosk has tampered

rule name	action	example
replace-well-known	Match name attribute in post-arg-tag or get-arg-tag element and replace value attribute of well-known place-holder value with real value	replace-well-known <post-arg name="CreditCard" value="MyVisa"> 1234 e.g., request before and after are: <post-arg name="CreditCard" value="MyVisa"/> <post-arg name="CreditCard" value="1234" />
replace-mapping	Match name attribute in post-arg-tag or get-arg-tag element and replace value attribute looking up mapping from Transient Store	replace-mapping <post-arg name="StockSymbol" value="MI"/> AOL e.g., request before and after are: <post-arg name="StockSymbol" value="MI"/> <post-arg name="StockSymbol" value="AOL" />
replace-form	Replace obfuscated request data with real data from Transient Store	replace-form <StockOrderCgiName>
replace-cookie	Replace obfuscated cookies	replace-cookies <DatekOnLine>

Figure 3: Content rewriting rules used by the FCM

with the data.

The *obfuscate-form* rule removes sensitive information encoded in form fields and hyperlinks. The rule generates temporary obfuscated names for form action names, form input fields, and hidden fields for state. Mappings between the obfuscated values and the actual data are held in the Transient Store. The rule also modifies sensitive information in hyperlinks in HTML pages that may contain user data as well as JavaScript. These transformations are necessary to prevent the leakage of sensitive data to kiosks, as well as to prevent kiosks from being able to post form requests directly to the end service. The *obfuscate-cookie* rule performs a similar transformation for any cookies that may be sent to the untrusted endpoint.

2.5.2 Using Content Rewriting to Reduce the Entry of Sensitive Information

Just as we do not trust endpoints to receive sensitive data, we do not want the user to enter sensitive information into untrusted kiosks. To handle these cases, the FCM uses replacement rules to exchange placeholder information with sensitive information that is stored only in the trusted infrastructure. There are four rules for content rewriting: *replace-well-known*, *replace-mapping*, *replace-form*, and *replace-cookie*.

The *replace-well-known* rule replaces well-known identifiers with the actual data, such as “CREDIT CARD”, with the user’s credit card number, and “HOME ADDRESS” with the user’s real home address. This type of form replacement is useful for supporting e-commerce purchase requests that need account numbers and shipping addresses that can’t be entered on untrusted clients. This functionality is also useful for trusted clients as it simplifies the process of filling out forms.

The *replace-mapping* rule replaces obfuscated values with their mapped values using information stored in the Transient Store by the *obfuscate-mapping* rule. For example, a user selling a stock enters the abbreviation for a U.S. state name rather than the stock symbol, and the FCM performs the reverse mapping.

The *replace-form* rule is used to replace obfuscated form action names, field names, and hidden text in HTTP GET/POST requests using mappings previously stored in the Transient Store. The *replace-cookies* rule works in similar fashion on cookies.

2.5.3 Using Control Filtering to Protect Service Functionality

Many services provide information with varying degrees of sensitivity. For example, looking up a stock quote is much less sensitive than trading stock. However, most services provide all or nothing access to service functionality, as they assume that the endpoint is trusted. To support access from untrusted endpoints the infrastructure must provide fine-grain access control over service functionality.

This capability is provided by the FCM, which uses control filtering rules to raise the security level such that users are comfortable performing sensitive actions. The FCM performs filtering operations on the control data flowing from the client to the server based upon matching XML representations of HTTP GET and POST requests.

The FCM uses two rules for control filtering: *allow-control* and *authorize*. The *allow-control* rule states whether or not a request to submit a form or

rule name	action	example
allow-control	Explicitly allow control requests for posting and getting pages	allow `<StockOrderCgiName>`
authorize	Request and wait for out-of-band authorization of control requests from trusted mobile device	authorize `<StockOrderCgiName>`

Figure 4: Control filtering rules used by the FCM

view a page can be transmitted to the end service. In the stock trading example, the control functions on the service's forms and pages (even in the split-trust case) must be explicitly allowed by the profile.

The *authorize* rule provides fine-grain control over sensitive actions, such as buying stock from an untrusted endpoint by using an a trusted device to explicitly authorize the request. In the stock trading example, a request to trade stocks is intercepted and held by the FCM until the authorization is received from the user's trusted companion device. In this example, a PDA is used to authorize requests to the service; however, alternatives include using devices such as cellular telephones or two-way pagers. For example, if users want actions to be authorized through their cellular telephones, they need to first register the phone with the infrastructure as the authorization device. Then, when an action is performed that requires confirmation, the FCM will intercept it, hold it, and call a user on her cellular telephone. The trade action is then read to the user and confirmation is requested. Once it is authorized, the FCM releases the request.

2.6 Identity Service (IS)

The Identity Service is the secure repository of persistent data associated with users' *trust profiles* and their security preferences for access to services. The Identity Service stores the following types of information:

- Identities and credentials (e.g., usernames, passwords, private keys, shared keys, etc.) for accessing secure services.
- Rules and preferences for content and control filtering.
- Personal information, such as addresses and account numbers for use in automated form filling from untrusted devices.
- Pseudonym identities and one-time password information.

The primary motivation for the Identity Service derives from the need to hide private data from untrusted endpoints, where there is no way for a user to securely transmit private information. The Identity Service provides an alternative where users' passwords and other private information are stored in the infrastructure and merged with any transmission after the data has left the untrusted device and entered the trusted infrastructure.

In addition to enabling access from untrusted endpoints, the Identity Service provides benefits for users accessing secure services from trusted endpoints. The replication of security credentials on several devices, some of which may easily be stolen (e.g., small PDAs), increases the probability that credentials may be compromised. It is also unlikely that most users will be savvy enough to understand the security capabilities of several devices, operating systems, and file systems. Similarly, changes to a credential require that the credential be updated on all devices. The IS solves these problems by providing a single source for such data.

Obviously, the security model of the Identity Service must be carefully considered and it must be well-protected against attacks. We argue that storing the data in the Identity Service is more secure than having it managed across several devices by end users. It is much easier to monitor and protect access to one professionally-administered, physically-secured service, than for naive users to administer several physically insecure devices.

Users have control over access to their private information in the Identity Service, as each of the data entries has an Access Control List (ACL) associated with it. The ACL allows users to specify which transcoders are allowed to access a specified data item.

Since the Identity Service centrally manages all user credentials, it is a requirement that the service be scalable and highly available. It is also crucial that the Identity Service be well managed and guarded against attacks. Denial of Service attacks on the Identity Service can be precluded by placing the IS behind a firewall. Overall, the user of

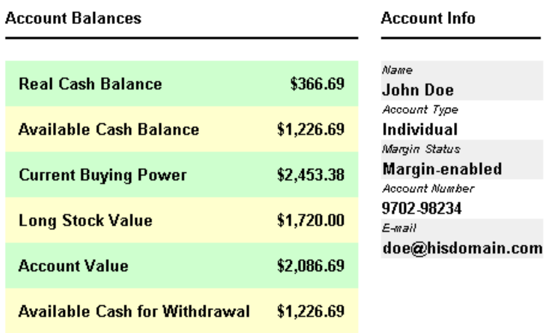

Figure 5: **Stock Trading Portfolio Page**

the untrusted device is better off having the Identity Service since the user avoids the current model of entering real passwords.

2.7 Transient Store (TS)

The Transient Store is used to store soft state associated with the current session of the user, such as mappings of obfuscated to original URLs, pending requests for trusted device authorization, or session information. Data can be stored using index keys that are generated by applying a secure hash function (e.g., SHA1 [28]). Secure access can be provided by using either a large key space such that a key can never be guessed or by performing access control.

3 Demonstration: Stock Trading

In this section, we illustrate, in detail, the relevant operations and messages between components in the stock trading example. We start by illustrating content filtering of the data flowing to an untrusted airport kiosk running a web browser. We follow with the more complicated control filtering procedure of buying stock with the aid of a trusted PDA. These illustrations use the Datek Online stock trading service. [11]

3.1 Content-Filtering

Using the stock trading example, consider the actions required to filter sensitive content from a page. For example, the users stock portfolio page shown in Figure 5.

The server-side Format Transcoder recognizes the type of page being sent back from the service,

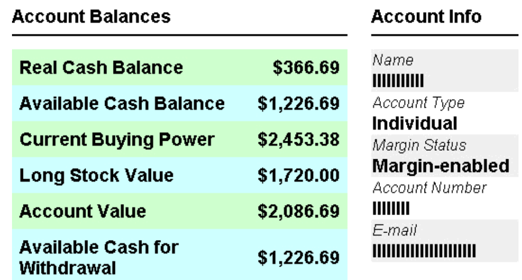

Figure 6: **Filtered Stock Trading Portfolio Page**

extracts the page's semantic data, and formats the data as an XML document. Consider, for example, the table in the right column of Figure 5 that contains sensitive personal information about the user. Such information would be marked up in XML as follows.

```
<Stock_Account_Info>
    <Name> John Doe </Name>
    <Account_Type> Individual </Account_Type>
    <Margin_Status> Margin-Enabled </Margin_Status>
    <Account_Number> 9702-98234 </Account_Number>
    <E-Mail> doe@hisdomain.com </E-Mail>
</StockAccountInfo>
```

The other semantic data on the page (e.g., account balance table, current positions, etc.) is similarly extracted into XML and presented to clients using an XSL style sheet for rendering.

In this example, the applicable rules are the *hide* rules associated with sensitive personal information[2]. The FCM uses the following content filtering rules to obscure valuable, sensitive personal data:

```
hide <Name>
hide <Account_Number>
hide <E-Mail>
```

The results of applying these rules are shown in Figure 6. Figure 7 displays the flow of messages and transformations performed by each of the components as the data for a page flows from the service to the user.

The stock trading service dynamically generates a portfolio page and sends it over an SSL connection into an infrastructure SSL Security Adaptor (1). The SSL Security Adaptor passes the web page's HTML to the server-side Datek Format Transcoder

[2] *Verify* rules for stock quotes and *replace-form* rules on the JavaScript anchors are also applicable, but are left out to simplify this illustration.

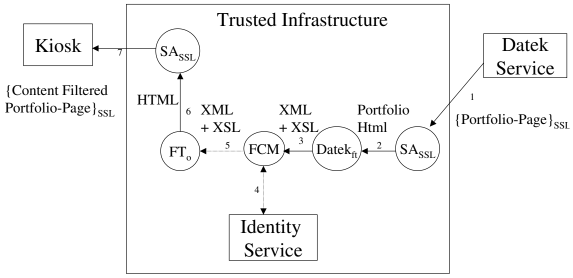

Figure 7: **Content Filtering Portfolio Page**

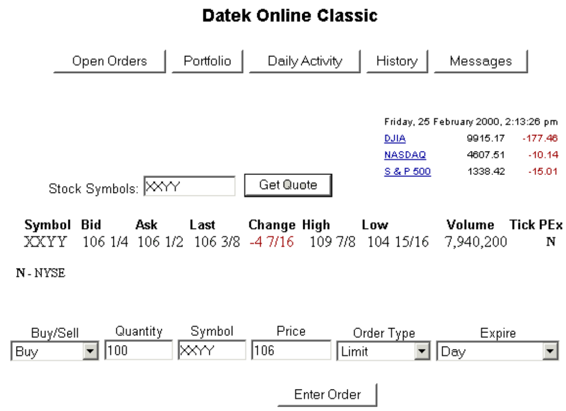

Figure 8: **Unfiltered Stock Order Entry Page**

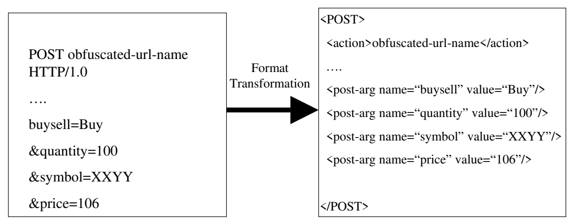

Figure 9: **POST request transformation**

(2), which translates the page into a XML-based semantic representation. The XML representation and associated XSL style sheet are passed on to the FCM (3). The FCM looks up the rules for the page based on the user's current profile (4), in this case the split-trust profile.

The FCM applies the content filtering rules to the data in XML format. Once the FCM has processed all the rules associated with the current page, it passes the modified XML and the XSL style sheet on to the client-side Format Transcoder (5), which renders the content as HTML for display on the kiosk. Finally, the HTML is passed to the client side SSL Security Adaptor (6), which sends it securely to the kiosk (7) over SSL.

3.2 Control Filtering and Leveraging the Trusted Portable Device

The previous content filtering example illustrates the functionality of each of the components and the flow of content down from the service to the untrusted device. Here we focus on the important aspects of control filtering as well as using a trusted device to verify and authenticate the user's requests. The user first enters an order to buy a stock from the kiosk as shown in Figure 8.

The incoming client-side Format Transcoder transforms the HTML form request into an XML representation as shown in Figure 9. The XML tag names and attributes represent HTML form argument names and values. The FCM first applies the *replace-form* rule to map obscured-url and form fields to their actual values. It does so by retrieving the mappings from the Transient Store. The FCM then applies the remaining rules to the operation. In this example, the only applicable rule is the *allow-control* rule[3], as the site generates a confirmation form as shown in figure 10 before placing the stock order.

Figure 11 illustrates the messages flowing up to the end service for a control filtering request. The user clicks the enter order button to submit the form for this request (1). The incoming Security Adaptor takes the data from the established SSL connection and passes it on to the incoming client side Format Transcoder (2), which transforms the HTTP POST request into an XML representation and then passes it to the FCM (3). The FCM retrieves the rules associated with the user's current role from the Identity Service (4) and uses them to process control requests. It does so by retrieving the mappings from the Transient Store (5). The modified XML representation of the POST request is passed to the out-going server-side Format Transcoder (6), which creates an HTTP POST request from the XML representation. The POST request is passed to the server-side Security Adaptor (7) which sends it securely to the Datek service (8) over SSL.

The order confirmation page is subject to content-filtering of sensitive data, as we mentioned previously. The more important action on this page is the control filtering that relies on the trusted mobile device to authorize the final stock purchase re-

[3]The *replace-mapping* rule could also be applicable, but it is left out of this example for simplicity.

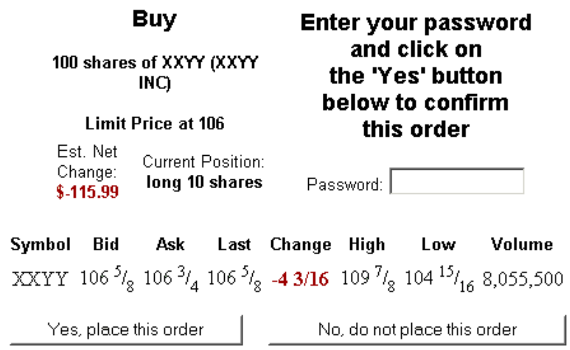

Figure 10: **Unfiltered Order Confirmation Page**

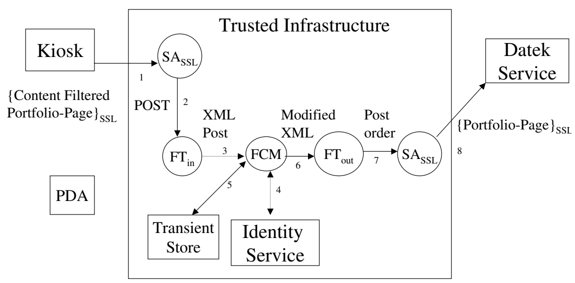

Figure 11: **Control Filtering Messages and Transformations**

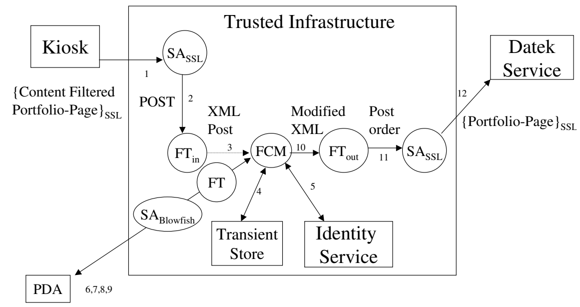

Figure 12: **Control Filtering with Trusted Device**

quest. The additional rules that apply to this page are *replace-well-known* and *authorize*. The replace-well-known rule is used to substitute the password "DATEK_PASSWORD" entered in the password field, with the user's true Datek password. The authorize rule requires the FCM to obtain authorization from the user's trusted device before allowing the submission to proceed.

Figure 12 shows the flow of messages using trusted device authentication. Steps 1 – 5 from figure 11 are repeated for the confirm order request page and its relevant fields. In this example, the FCM obtains authorization from the user's PDA using a previously established channel consisting of a PDA Format Transcoder (to render from the XML representation to the device format) and a Blowfish Security Adaptor.

The user uses the PDA's browser to request the authorization information page (6). The FCM generates XML that is rendered appropriately by the FT's for authorization on the PDA (7). The page displays verification data (stock name, price, etc.) and form buttons for authorizing and declining the request. The user then verifies that the information matches that entered on the untrusted endpoint. If it is acceptable, the user sends a message (8) authorizing the request and receives the response (9) on the PDA. Finally, the FCM releases the original POST action to the Datek service (10, 11, 12).

4 Current Status

The design we propose in this paper is a generalization of the ideas we gained from an initial prototype implementation of the system. The prototype provides access to the Datek Online Stock Trading service to users of untrusted kiosks and computationally weak PDAs.

Our implementation includes a prototype Identity Service and application-specific instances of Format Transcoders, FCMs, and Security Adaptors. It is written as a collection of Java services on a scalable, Internet service platform [3], and a Palm Pilot III application written in C++. We leverage a secure version of Java Remote Method Invocation (RMI) and a public key infrastructure to provide secure communication between components inside our environment.

The prototype Identity Service is implemented as a Java service with a secure RMI interface. The prototype Identity Service supports username/password and shared-key credentials. Entries in the prototype Identity Service are protected by ACLs that must be set appropriately for use by each of the other components in the framework. A web interface supports the insertion of user information as well as the management of access control lists.

When we evaluated the prototype implementation, we realized that it would be useful to generalize from storage of security credentials to other sensitive personal information. This allows more powerful replacement rules to support e-commerce transactions.

The prototype Security Adaptor supports proxied access to SSL services via a Blowfish challenge response authenticated socket from the Palm Pilot III. During this exchange, symmetric keys are exchanged for use as session keys. Our secure communication uses HMAC-SHA1 [28] for authentication of the endpoints and integrity, and 128-bit Blowfish [27] for encryption. Four session keys are generated: one encryption key and one message authentication cipher (MAC) key for each direction. Also, the MAC's are chained to prevent replay and splicing attacks. This protocol can establish a secure connection to a Palm Pilot III in 1.9 seconds. This is in comparison to doing the public key operations for SSL. In [5], the authors report that signing/decrypting a message on a Palm Pilot Professional takes about 5 seconds with a 512 bit RSA key and about 25 seconds with a 1024 bit key.

We have implemented two FCM's for access to the Datek Service. The kiosk FCM is accessed via an SSL socket, and authenticates the user with a simplified one-time password mechanism. The rules for content and control filtering are hard coded in the kiosk FCM. The PDA FCM scrapes the Datek data from the portfolio page and presents it to the user as simple text. The Palm Pilot III application uses Blowfish sockets and displays the scraped text from the Datek Page. We also have initial experience with a WinCE client that uses a lightweight version of the RMI protocol into a proxy in the infrastructure for trading stocks from this device.

In evaluating the prototype implementations of the FCM and Security Adaptors, we learned several important things. Firstly, a filtering proxy with preestablished rules is of limited value for interacting with secure services. The amount of useful functionality that can be exposed to an untrusted endpoint with reasonable security is minimal. Experience with the PDAs quickly showed the drawbacks in using them to initiate all operations. This led us to the fusion of multiple devices, and use of the out-of-band channel to the trusted device for authorization of sensitive operations. Secondly, fine grain content and control filtering are powerful transformation concepts, but they operate on a semantic level rather than a structural level used by most proxy transformations. We found that making the task of extending the service to multiple end-services and different client devices tractable required the separation of semantic data from presentation data, and rule-based filtering operations on the semantic data.

5 Related Work

Several related projects have focused on data filtering at different levels of granularity for increased security. The research that is most related to our work is presented in [35]. This project examined the problem of protecting the release of data from databases for collaboration purposes, where the internal databases are not organized according to external access criteria. Wiederhold and Billelo note that, in these environments, traditional authentication and authorization tools and secure transmission fail to protect the release of inappropriate data.

They propose a solution based upon a security-mediator that uses a set of rules to perform result-checking that filters both queries and, more importantly, their results. Their rules for filtering query results contain traditional role-based access control, site blocking, and more advanced features, such as the replacement of query result components with non-identifying (obfuscated) terms. We use this type of functionality in the filtering and content-modifying component of our architecture to protect against the inappropriate release of data at an untrusted public endpoint.

Our architecture's approach to content and control filtering is based upon having specific semantic knowledge of the service data and control actions that are being filtered or modified. However, such an approach does not work for protecting unstructured information, such as the content of e-mail messages. Instead, we believe that we could apply techniques from medical research that focus on generalizing, substituting, and removing information from medical data to protect privacy without obscuring "important" details of the data [30, 31].

Related to our idea of multiple device fusion is the notion of *splitting trust* between a PC and a mobile device to increase security [5]. In the split trust paradigm, the application's security crucial parts are run on a small trusted device, while the rest of the application executes on a more powerful but untrusted device.

The authors explore the split trust paradigm in the context of an e-mail application that uses a Palm Pilot to perform encryption and decryption operations, as a PKCS11 smart card, for Netscape Communicator. The application enables users to view encrypted messages on the Palm Pilot. However, as the authors point out, this powerful paradigm has several shortcomings. First, the untrusted endpoint can still forge e-mail messages by displaying one e-mail on screen while asking the PDA to sign a different e-mail. Unless the message being signed is displayed on the PDA, one can not guarantee that the signature is associated with the right message. Second, the untrusted endpoint still has access to unencrypted e-mails. To address these problems, our architecture includes components for dynamically splitting trust and functionality. It combines user preferences with an out-of-band channel for sending authorization messages to the PDA, and also determines the content that is displayed on the untrusted device.

Another related project is the work by the World Wide Web Consortium on the Platform for Privacy Preferences (P3P) [9, 10] and the associated preference exchange language – APPEL [18]. The goals of the P3P platform are to promote privacy and trust on the web by providing service providers with a means to disclose their practices regarding the collection and dissemination of personal information and providing users with the ability to make informed decisions regarding the use of this information. Web sites present their privacy policies and the users' agent makes semi-automated decisions based on the users' privacy preferences. This research is primarily focused on protecting the information sent from users to services, varying from click-stream data to users' personal information, such as names and addresses. Our research explores the inverse of this problem: allowing users to express their preferences about the type of data to be displayed and the functionality to be allowed as a function of the users' access device. Also related to this research area are trust-management systems, such as KeyNote [6], that serve as engines for applications to determine whether or not a potentially dangerous operation conforms to a user's security policy.

Our infrastructure-based proxy model is also related to research on proxies to more efficiently use network resources, reduce cost, and increase security focusing on filtering to optimize performance along the last link in mobile environments [37, 38, 2]. Their filtering operations operate at both the application level (e.g., selectively drop MPEG frames or reduce bandwidth to X) and at the protocol level (e.g., delaying data, compressing data, or acting on a client's behalf to reduce traffic). They stress the ease of deploying an intermediary instead of changing the protocols at end points. Their proxy execution environment consists of both interpreted rules and native code for increased performance filtering at lower levels. In addition, the proxy is a trusted part of the platform that is capable of modifying users' data and requires users to authenticate to the proxy. We believe that a similar argument can be applied to the delivery of content to untrusted endpoints. Our architecture operates as a layer above the application layer, performing semantic filtering of the content.

Related systems have examined the problem of managing different types of identity data; for example, Apple's KeyChain [4] stores all username-password pairs locally on the client PC. Lucent's web proxy, ProxyMate [19], takes this capability a step further by generating stronger user ID's and passwords transparently to the user, and using them to login to the actual site. Novell's DigitalMe [24] and Yodlee [36] store passwords on a central server[4]. Microsoft's Passport [22] service as well as other e-wallet services hold personally identifying information to simplify form filling in e-commerce transactions. This functionality is similar to our architecture; however, we make such information always available by keeping it in the infrastructure, rather than on a user's end device. CommerceNet's IdentitySafe [8] provides users with pseudo identities for purchasing from online merchants and only releases personally identifiable information to sites on a need to know basis.

Our application of security protocol adaptation is derived from the ideas presented in [14], which suggested that security transformations for small devices would be desirable and presented proxy-based access to Kerberized services. Their design places the bulk of Kerberos authentication work on an infrastructure-based proxy, providing indirect authentication of clients and simplifying the requirements of small devices.

Finally, to format content for a range of devices with limited display capabilities, we could leverage related research, such as TACC [15, 16], to format the content to these devices based on the network and device capabilities. By abstracting device functionality into the client-side Format Transcoder, we simplify the process of adding new types of devices. In addition, the Wireless Application Protocol(WAP) [33] presents a standard format for small devices. Other related work in this area includes Oracle's Portal-To-Go which promises any service to any device access. [25]

[4]These systems provide convenience, but do not enhance security from untrusted endpoints, as the master password still needs to be entered and thus, can be captured or hijacked by the untrusted endpoint. In contrast, our architecture's Identity Service provides one-time passwords and fine-grain control over security.

6 Future Work

We are in the process of extending the prototype implementation to encompass the architecture presented in this paper. This process involves separating the components we have bundled together into cleanly separated layers, as proposed here. We are investigating the markup of the Datek Service semantically in XML with XSL for the presentation logic and a rule-driven FCM. Additionally, we are investigating support for split-trust authentication in the FCM.

Our next step is to push forward on incorporating new devices and services into our prototype implementation. This step will help to verify the extensibility claims that we have made and it will help us revise and improve our FCM model. Also worth looking at is applying the concepts in the framework to other Internet protocols such as for e-mail (IMAP) or directory services (LDAP).

There are several new and interesting applications enabled by our framework. For example, the Identity Service can easily support short-lived pseudonyms. Let's say you want to ask the pizza delivery person to send you an instant message just prior to his arrival; however, you do not want to give him your real identity (to avoid receiving spam in the future). The Identity Service can generate a short-lived identity that expires in a few hours or after several uses. Accomplishing the equivalent action in the current world requires the user to go through a cumbersome process of creating an account and deleting it later.

The Identity Service can also support persistent pseudonyms that enable users to establish web presences without revealing their real identities. Placing the pseudonym support in the infrastructure permits pseudonymous access to services from any device.

The Identity Service also allows software agents to act on a user's behalf. For example, an automated stock monitoring system could watch your portfolio and notify you if significant events occur but could not initiate stock trades [34].

7 Conclusion

In this paper, we present a novel infrastructure-based architecture for providing secure, wide-area access to services. The architecture enables access from a wide variety of end devices by using content and security transformation functions with rule-driven filtering to decouple device capabilities from service requirements. This approach greatly simplifies and reduces the amount of work that must be done to support a new device or service. We provide a generic "any device to any service" model that is in stark contrast to the traditional approaches of vertical service and device integration and implementation. Our hypothesis is that such a generic model provides rapid support for new devices and services, a critical requirement for the Post-PC era.

Furthermore, the architecture also supports a new model of secure interaction from untrusted public Internet access points found in mobile environments. To our knowledge, our architecture is the first to address this interaction model. By providing a generic control and content rewriting capability, the architecture provides users with precise control over the exposure of information, both as a function of their access devices and as a function of the users' adopted roles. Additionally, the architecture supports the fusion of multiple devices, using trusted portable devices for secure authorization of sensitive requests.

References

[1] 3Com. Web Clipping Applications Tutorials. http://www.palm.com/devzone/webclipping, January 2000.

[2] Elan Amir, Steve McCanne, and Hui Zhang. An Application Level Video Gateway. In *Proc. ACM Multimedia 95*, San Francisco, CA, 1995.

[3] Anonymous. Suppressed for anonymity. In *Anonymous Proceedings*, 1999.

[4] Apple. Keychain. http://www.apple.com/macos/feature4.htm, January 2000.

[5] Dirk Balfanz and Edward W. Felten. Hand-held computers can be better smart cards. In *Proceedings of the Eighth USENIX Security Symposium (Security '99)*, Berkeley, CA: USENIX Association, August 1999.

[6] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. RFC 2704: The KeyNote Trust-Management System Version 2. http://www.crypto.com/papers/rfc2704.txt, September 1999.

[7] Certicom. Elliptic Curve Cryptography for Palm VII. http://www.certicom.com/press/98/dec0298.htm, December 1998.

[8] CommerceNet. IdentitySafe. http://www.commerce.net/project/hotsheet.html, January 2000.

[9] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler, and Joseph Reagle. Platform for Privacy Preferences (P3P1.0) Specification. http://www.w3.org/TR/P3P, 2000.

[10] Lorrie Faith Cranor. P3P Guiding Priniciples. http://www.w3.org/TR/1998/NOTE-P3P10-principles, 1998.

[11] Datek. Datek OnLine Trading Service. http://www.datek.com, January 2000.

[12] Security Dynamics. SecurID. http://www.securitydynamics.com/products/securid, January 2000.

[13] Aharon Adler et al. Extensible Stylesheet Language (XSL). http://www.w3.org/TR/xsl/, 2000.

[14] Armando Fox and Steven D. Gribble. Security on the Move: Indirect Authentication using Kerberos. In *Proceedings of the 2nd ACM International Conference on Mobile Computing and Networking (Mobicom '96)*, Rye, New York, November 1996.

[15] Armando Fox, Steven D. Gribble, Eric A. Brewer, and Elan Amir. Adapting to client variability via on-demand dynamic distillation. In *Proceedings of the ACM Seventh International Conference on Architectural support for Programming Languages and Operating Systems*, Cambridge, Massachusetts, October 1996.

[16] Armando Fox, Steven D. Gribble, Yatin Chawathe, and Eric A. Brewer. Scalable network services. In *Proceedings of the 16th ACM Symposium on Operating Systems Principals (SOSP-16)*, St. Malo, France, October 1997.

[17] A. Frier, P. Karlton, and P. Kocher. The SSL 3.0 Protocol. http://www.netscape.com/eng/ssl3/ssl-toc.html, March 1996.

[18] Marc Langheinrich. A P3P Preference Exchange Language (APPEL) Working Draft. http://www.w3.org/TR/WD-P3P-preferences, 1998.

[19] Lucent. Proxymate. http://www.proxymate.com/, January 2000.

[20] N. Maller. The S/KEY One-Time Password System. http://www.ietf.org/rfc/rfc1760.txt, February 1995.

[21] N. Maller, C. Metz, P. Nesser, and M. Straw. A One-Time Password System. http://www.ietf.org/rfc/rfc2289.txt, February 1998.

[22] Microsoft. Passport. http://www.passport.com, January 2000.

[23] mySimon Inc. mySimon inc. Backgrounder. http://www.mysimon.com, January 2000.

[24] Novell. Digitalme. http://www.digitalme.com/, January 2000.

[25] Oracle. Oracle Portal-to-Go Any Service to Any Device. http://www.oracle.com/mobile/panbwp.pdf, October 1999.

[26] Palm.com. Palm V PDA. http://www.palm.com/products/vseries.html, 1999.

[27] Bruce Schneier. Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). In *Fast Software Encryption, Cambridge Security Workshop Proceedings*, December 1993.

[28] Bruce Schneier. *Applied Cryptography*. John Wiley and Sons, Inc., 1996.

[29] Atsushi Sugiura and Yoshiyuki Koseki. Internet scrapbook: automating Web browsing tasks by demonstration. In *11th annual ACM symposium on User interface software and technology*, San Francisco, CA, 1998.

[30] Latanya Sweeney. Replacing personally-identifying information in medical records, the Scrub system. In *Proceedings of the American Medical Informatics Association.*, Washington, DC: Hanley and Belfus, Inc, August 1996.

[31] Latanya Sweeney. Guaranteeing anonymity when sharing medical data, the Datafly system. In *Proceedings of the American Medical Informatics Association Symposium.*, Washington, DC: Hanley and Belfus, Inc, August 1998.

[32] C. M. Sperberg-McQueen Tim Bray, Jean Paoli. Extensible Markup Language (XML) 1.0. http://www.w3.org/TR/REC-xml, 2000.

[33] WAP. WAP Forum Specifications. http://www.wapforum.org/what/technical.htm, January 2000.

[34] J. E. White. Telescript technology: The foundation for the electronic marketplace, 1994.

[35] Gio Wiederhold and Michel Bilello. Protecting Inappropriate Release of Data from Realistic Databases. In *Proc. of Data and Expert Systems (DEXA) Security Workshop*, IEEE, August 1998.

[36] Yodlee. Yodlee. http://www.yodlee.com/, January 2000.

[37] Bruce Zenel and Dan Duchamp. General purpose proxies: solved and unsolved problems. In *Proceedings of the Sixth Workshop on Hot Topics in Operating Systems*, Los Alamitos, CA, USA: IEEE Comput. Soc. Press, 1997.

[38] Bruce Zenel and Dan Duchamp. A general purpose proxy filtering mechanism applied to the mobile environment. In *Proceedings of the Third Annual ACM/IEEE Conference on Mobile Computing and Networking (Mobicom '97)*, New York, NY USA: ACM, 1997.