Using Trustworthy Computing to Enhance Privacy

Chris Karlof Yaping Li Emil Ong
University of California, Berkeley
{ckarlof,yaping,emilong}@cs.berkeley.edu

***Abstract*— Privacy in digital systems is currently regulated by the law and policy, but individuals still see their personal data abused. We believe individuals should have a technical mechanism which allows them greater control over their personal information. We propose the use of *software agents* to increase privacy in the digital world. Software agents are small pieces of software that digitally represent an individual and protect her personal information. In this paper, we will show a novel use of *trustworthy computing platforms* to construct the agent architecture. We develop two motivating examples for using agents: Ecommerce and reputation logs. Finally, we discuss the challenges in this approach and conclude with a summary of what functionality is gained over current solutions.**

I. Introduction

Privacy is the ability to control one's personal information. If we consider an individual's personal information to be her name, her address, and other small pieces of textual data, then individuals currently do not have much privacy in the digital world. We believe individuals should have a technical mechanism which allows them greater control over their personal information. We investigate exploiting *trustworthy computing platforms* for this purpose. These platforms are the product of a recent trend in computer and operating system manufacturing which allow a third party to understand, verify, and to some extent control a platform which they do not own. To provide greater privacy to individuals we propose the use of software agents, small pieces of software that digitally represent an individual, on trustworthy computing platforms.

II. Problem Statement

Massive databases of personal information have become pervasive in organizations and corporations, and many individuals are not fully aware of the extent to which this collected data can ultimately be used. Some examples include:

- Credit cards and discount cards are used to track the purchases and location of customers.
- Websites use cookies to track the browsing behavior of users.
- Electronic highway toll payment systems used to monitor traffic flow can be used to track the movements of a specific individual.
- Passive means of surveillance such as face recognition are being deployed at airports and sporting events to identify suspected criminals and terrorists [1].

These databases by themselves pose significant risks to personal privacy, but data mining techniques can combine and correlate two or more databases to reveal even more information about individuals. For example, two companies could share data on the purchasing history of common customers in order to better target marketing campaigns. A new U.S. government program, called "Total Information Awareness," [2] correlates data in various databases (commercial and governmental) to find potential terrorists.

For some individuals, this collection and correlation of personal information may only lead to an increase in annoying unsolicited phone calls and emails. However, aggressive profiling can have more serious consequences, causing an individual to be embarrassed in the home or workplace [3] or even be falsely suspected of criminal activity [4], [5].

There have been several attempts to protect individuals against these dangers via the law [6]. These laws are often well meaning, but have numerous problems:

1) Laws are limited to their respective jurisdictions; they may differ greatly from country to country and region to region.
2) Laws are frequently written ambiguously, sometimes requiring lengthy court proceedings to clarify their meaning.
3) Few individuals have the time, knowledge, or willingness to fully comprehend the scope and meaning of many laws and how an organization's privacy policy relates to those laws.
4) Different laws are often based on different definitions of privacy.
5) Individuals are rarely in a position to detect abuses. Even if they are able to discover a violation, prosecution has proven difficult [7].
6) Preventative measures against (external or internal) privacy violations are rarely stipulated.

These drawbacks lead to an unfortunate implication: *individuals largely have to trust an organization to respect its own privacy policy.* This is entirely unacceptable. Individuals need more consistent, more uniform mechanisms for gaining assurance about the use of one's personal information.

III. Personal Information Control (PIC)

We believe the gross shortcomings of privacy legislation can be overcome by appropriate *technical* mechanisms for protecting personal information. We define a *Personal Information Control* (PIC) system to be a technical mechanism that gives an individual direct control over the use of her personal information regardless of where the information is stored. The two key goals of a PIC system are *uniformity* and *assurance*.

A PIC system should provide *uniformity* in sense of empowering individuals to determine themselves how their personal information is used and not be subject to varying instances and interpretations of privacy policies and the law. More concretely, a PIC system should enable an individual to specify to which organizations her information can be released and in what manner that information can be used. She should be able to specify this policy once and have it apply to all organizations, or be able to specify completely different usage policies for different organizations. Usage policies should also be malleable. An individual should be able to alter her usage policy for any organization at any time.

A PIC system should provide *assurance* in the sense of guaranteeing the proper enforcement of one's usage policy. This entails not only preventing unauthorized access to and use of one's personal information, but also enabling an individual to monitor valid access and use.

We include the identity of an individual in the set of his or her personal information. This inclusion means that if someone wishes to remain anonymous, a PIC system should support that. Moreover, the ideal system would support anonymity along side of all the other requirements, including notification.

We make one major assumption regarding PIC systems. Since we are mainly concerned with data in an electronic setting, we assume all data remains contained within digital systems (usually computers). Notice that while this seems to be an obvious assumption, it is a very important one – *a technical enforcement mechanism is unlikely to be effective once data escapes the digital domain*. "Analog holes" such as physical copies (e.g. printouts) or data displayed on a monitor represent a significant threat to the confidentiality of personal data. The best a PIC system can do is try to minimize the necessity of such situations.

IV. Digital Rights Management (DRM)

By viewing personal information as the content of individuals, the goals of Personal Information Control are quite similar to those of Digital Rights Management (DRM). DRM systems were originally intended to help content owners securely distribute digital media to a large number of consumers in a manner that protects the interests of the owner. An effective DRM architecture should both prevent unauthorized duplication of content as well as enable the owner to restrict and monitor how the content is used.

Although it may seem intuitive to try to adapt existing DRM techniques to develop a PIC system, DRM and PIC still have several fundamental differences. First, DRM is generally intended to manage relatively large instances of multimedia such as collections of songs or images, feature films, or electronic books. Although it is conceivable that some personal information might be stored as audio or video, the large majority is simple ASCII text. Second, in DRM systems, the relative number of content owners is small compared to the number of consumers. In PIC, it is the reverse. This reversal suggests PIC systems will face scalability issues that traditional DRM systems were not designed to handle. Third, the resources of content owners in DRM and PIC systems are highly mismatched, and this mismatch will constrain the implementation options available to PIC systems. For example, a media company might distribute smart cards and readers to enable authorized consumers to view content, but it is clearly not feasible for an individual to do the same to authorize an organization to access to her personal information. Finally, the costs and ease of illegal redistribution of content in both systems is significantly different. A pirate redistributing an illegal copy of a feature film must absorb non-negligible costs in terms of bandwidth or physical media and equipment. In contrast, an unethical organization (or employee of an organization) can sell information on millions of individuals by transferring as little as a few gigabytes. Even worse, the privacy of a small number of individuals can be compromised by pencil and paper if the adversary can simply display their information on a monitor.

For the abovementioned reasons, it is unlikely traditional DRM techniques like watermarking, fingerprinting, code obfuscation, serial numbers, license managers, or proprietary hardware and software will be useful in developing PIC systems. Despite the fact that most of these technologies have been largely unsuccessful in DRM systems on their own right, they were developed in the context of large, complex media files, few owners and many consumers, and a highly resourceful distribution infrastructure, and it is unclear how they can be easily adapted for use in PIC systems.

A more serious issue with using DRM for PIC systems is the fact that overall DRM deployment has been quite sparse. Content owners have been reluctant to adopt DRM systems largely because deployment is currently limited to untrustworthy general purpose computing platforms. We refer to the vast majority of today's machines as "untrustworthy" because it is difficult for a party who is not the owner of the machine to gain assurance about the behavior of the machine while it is in possession of a specific program or collection of data. More concretely, it is difficult to prevent an untrustworthy computer from running a program under a debugger, searching for cryptographic keys hidden within binary code, or simply copying and redistributing a file. Untrustworthy machines in this setting are assumed to be outright hostile, and the large majority of proposed DRM solutions fail when they are deployed on hostile, untrustworthy computing platforms [8], [9], [10], [11].

V. Trustworthy Computing

These past failures are the motivation for *trustworthy computing platforms*. Trustworthy computing platforms are systems whose behavior can be understood, verified, and to some extent controlled by third parties. Trustworthy computing is believed to be a key enabling technology for widespread deployment of DRM systems, and various media companies and computer system manufacturers are interested in hastening its development and acceptance [12], [13].

Trustworthy computing holds great potential for DRM systems, but there is nothing immediately obvious that would prevent it from being used in PIC systems as well. As easily as content owners can gain assurance about how a consumer's system handles copyrighted digital media, individuals could gain assurance about how an organization's system handles sensitive personal information. However, its ultimate success is not clear either: the key differences between PIC and DRM enumerated in the previous section remain. Herein lies the main focus of our paper: *How effectively can trustworthy computing platforms be used to implement Personal Information Control systems?*

A. Trustworthy Computing Components and Primitives

In order to evaluate the effectiveness of trustworthy computing platforms in enhancing privacy, it helps to first understand their basic components and exported primitives.

1) Hardware/software Components: Two essential hardware/software components are the *trustworthy platform module* (TPM) and the *trusted root*. The TPM is a tamper-resistant (and possibly tamper-responsive) hardware module that exports a range of cryptographic functions and is able to measure and record various information about the current hardware and software state of the system. Each TPM has a unique *endorsement key*, a public/private keypair used for signatures that is signed by the system manufacturer's private key in a certificate. The endorsement key is the root of trustworthiness in the system, and it is vital that the private component of it is not known by any other principal, including the system owner.

The trusted root provides two essential functions: it mediates requests for cryptographic operations to the TPM and provides guarantees to applications regarding the current state of the system. With respect to the latter, the trusted root is essentially a secure memory manager. It not only provides strong memory isolation, but also enables an application to limit what other applications are running concurrently with it. The trusted root can also be thought of as a trustworthy component of a larger untrusted operating system. The trusted root may interact with untrusted OS at times, reading and writing files or requesting disk space, for example.

A trusted root might use the TPM to measure the current hardware configuration and export that information to applications as well. For example, a media player for digital movies might request that the video card is trusted and it is executed only when programs it trusts are executing concurrently on the system. Concurrently running screen capture programs or a video card which captures and exports video images before sending them to the screen (which a media player would certainly not trust) would be detected by the trusted root and the media player would be prevented from running. A cryptographic hash of the currently running trusted root is measure at boot time and stored within the TPM.

2) Basic Primitives: Three basic primitives useful in trustworthy computing platforms are *outbound authentication* [14], *secure execution paths*, and *sealed storage*. Outbound authentication serves two purposes: it authenticates the current state of the system to a remote third party and enables the establishment of secure channels between the TPM and that party. Typically, the minimal amount of information a third party needs to gain assurance about the system is the TPM's endorsement key certificate, a statement of the current hardware configuration, and hash of the currently running trusted root. All of this information is signed by the endorsement key. An example of such an protocol between the TPM running on system B and a third party A is shown below:

$$\begin{aligned}
A \rightarrow B \quad &: \quad N_A, Cert_A && (1)\\
B \rightarrow TPM \quad &: \quad N_A, B, Cert_A && (2)\\
TPM \rightarrow B \quad &: \quad Cert_{TPM}, [B, N_A, N_{TPM}, HC, && (3)\\
&\quad\quad H(TR), \{N_A, N_{TPM}, K_s\}_{K_A}]_{K_{TPM}^{-1}}\\
B \rightarrow A \quad &: \quad Cert_B, [N_A, N_B]_{K_B^{-1}}, Cert_{TPM}, && (4)\\
&\quad\quad [B, N_A, N_{TPM}, HC, H(TR),\\
&\quad\quad \{N_A, N_{TPM}, K_s\}_{K_A}]_{K_{TPM}^{-1}}
\end{aligned}$$

where HC represents the current hardware configuration of B, TR is the trusted root currently running on system B, H is a cryptographically secure hash function, K_s is a fresh symmetric key generated by the TPM, K_A is the public key of A, and K_{TPM}^{-1} is the endorsement key of the TPM. We assume $Cert_{TPM}$ is signed by the TPM's manufacturer, and $Cert_A$ and $Cert_B$ are signed by a mutually trusted Certificate Authority attesting to both's identity. We also assume $Cert_B$ asserts that the TPM is indeed owned by B, and this assertion is signed by both the TPM and the mutually trusted CA. Our final assumption is, of course, that the TPM has not been compromised by B.

At the end of this protocol, 1) A gains assurance about the current hardware and software configuration of B, and 2) A establishes a secure channel with the TPM.

After a remote party A has established K_s with the TPM, and it is satisfied with the current trusted root and hardware configuration on B, it can create a *secure execution path* (SEP) on B. A SEP enables a remote party to install an application containing sensitive information such as secret keys or industrial secrets on a trustworthy computing platform that is bound to a particular TPM, trusted root, and hardware configuration. A creates a SEP by sending the following messages to the TPM and the trusted root TR via B:

$$\begin{aligned}
A \rightarrow B \rightarrow TPM \quad &: \quad [\{K, A, N'_A, HC, H(TR), && (5)\\
&\quad\quad Cert_B\}]_{K_s}\\
A \rightarrow B \rightarrow TR \quad &: \quad [A, N'_A, H(TR)]_{K_s}, [\{C\}]_K, D && (6)
\end{aligned}$$

where K is a fresh key generated by A, C is application code from A, D is any initial data, and N'_A is a fresh nonce. D is not protected under K, but rather by application specific keys stored in C.

The trusted root then requests disk allocations for the application code and its metadata MD_C from the untrusted OS. The trusted root also requests disk allocation for the application data D and its metadata. We assume this metadata contains sequence numbers or timestamps which allow the application or trusted root to verify that application data is not replaced with older versions. TR then sends the TPM[1] information about the allocated disk storage:

$$TR \rightarrow TPM \quad : \quad [A, N'_A, H(TR)]_{K_s}, MD_C, MD_D \quad (7)$$

[1]We assume a secure channel between the trusted root and the TPM. This may be a secure channel between memory and the CPU, a tamper-resistant enclosure, or a tamper-resistant memory and bus architecture.

The TPM validates the following information:

1) The trusted root and hardware configuration specified by A are indeed present.
2) The certificate for B is valid.
3) The N'_A and $H(TR)$ fields received in messages from A and TR are the same.

TPM then sends the following record back to TR :

$$TPM \rightarrow TR \quad : \quad A, [\{K, A, MD_C, MD_D, HC, H(TR), Cert_B\}]_{K^{SEP}_{TPM}} \tag{8}$$

where K^{SEP}_{TPM} is a symmetric key generated by the TPM for guaranteeing confidentiality and integrity of the application keys K used in SEP storage. K^{SEP}_{TPM} is stored within the TPM and is not known by any other principal. TR then stores

$$[\{C\}]_K \tag{9}$$

in the allocated regular disk area and

$$A, [\{K, A, MD_C, MD_D, HC, H(TR), Cert_B\}]_{K^{SEP}_{TPM}} \tag{10}$$

to a special SEP storage area. The trusted root relies on the TPM to maintain the root of a Merkle hash tree to guarantee the overall integrity of SEP storage. To guarantee integrity of application code and data regions, MD_D and MD_C could contain roots of Merkle hash trees to protect those regions as well. The trusted root would then provide an interface to verify the integrity of data after it has been read into memory. For higher assurance, the integrity of the memory should be guaranteed as well, but we do not consider that here.

Now, when a request arrives at the trusted root to execute the application installed by A, the TR forwards the request to the TPM along with its record in SEP storage:

$$TR \rightarrow TPM \quad : \quad A, [\{K, A, MD_C, MD_D, HC, H(TR), Cert_B\}]_{K^{SEP}_{TPM}} \tag{11}$$

The TPM verifies the current hardware configuration and trusted root match the entries in the record, and B's certificate is still valid. The TPM sends the metadata for A back to the trusted root:

$$TPM \rightarrow TR \quad : \quad A, MD_C, MD_D \tag{12}$$

The trusted root uses the TPM (which now has K) to decrypt and guarantee the integrity of the application code C. If valid, A's code C is loaded into memory, passed the metadata MD_D of its data section, and executed. At the beginning of its execution, A will likely check the integrity of its data D with keys stored in the application code or verify its Merkle hash tree.

Once an application A has a secure execution path, it can use *sealed storage*. Sealed storage is simply the ability of an application to securely store encrypted data on local durable storage. The principal challenge of this on untrustworthy systems is that encryption keys must be stored in the clear (or at least obfuscated) in the application code. Since applications with a secure execution path are stored encrypted on durable storage and only decrypted in memory, application encryption keys can be stored with application code with high assurance they will not be compromised[2]. With sealed storage, applications are able to keep its data secret from other applications as well as the platform owner.

B. Implementations of Trustworthy Computing Platforms

Proposed trustworthy systems have generally fallen into one of two types: *integrated trustworthy platforms* and*COTS secure co-processors*.

1) Integrated trustworthy platforms: The above approach for building a trustworthy computing platform is similar to current commercial ventures such as the Trusted Computing Platform Alliance (TCPA), Microsoft's Palladium, and the Digital Rights Management Operating System (DRMOS) [12], [13]. We refer to these approaches as *integrated trustworthy platforms*. The TPM is currently a tamper-resistant hardware module attached to the motherboard, but it is foreseen to eventually be designed as part of a tamper-resistant CPU. As the machine boots, the TPM measures and records the current hardware configuration and is responsible for loading and measuring the trusted root. After that, control is passed on to the operating system.

2) COTS secure co-processors: Another approach to building a trustworthy computing platform is to attach commercial-off-the-shelf secure co-processors to otherwise untrustworthy computing systems, typically via the PCI bus. Secure co-processors are essentially small trustworthy computing platforms co-located within a larger untrustworthy system. An example of device of this type is the IBM 4758 [15]. The IBM 4758 includes a Intel 486 microprocessor, several megabytes of RAM and FLASH, cryptographic hardware support, and a clock, all packaged in a tamper-responding enclosure. The 4758 is tamper-responding in the sense that it will zeroize internally stored secrets and other sensitive data when it detects attempts to compromise it or the secrets stored within. Attacks it currently detects include attempts to penetrate the enclosure, freezing, heating, applications of ionizing radiation, and variations in the supply voltage [16].

The 4758 can be programmed in C and uses an embedded operating system, CP/Q++. Although the computational capabilities of the 4758 are significantly less than current workstation class machines, it still provides a useful general purpose computing platform. The internal software architecture is divided into four layers with each layer protected from the higher ones via hardware. Layer 0 and Layer 1 boot the 4758 and perform self-tests to bring it into a secure state. Layer 2 is intended to be the operating system, and Layer 3 is for applications [15]. Lower layers sign certificates attesting to the state of the higher layers, with Layer 0's certificate signed by IBM. The Layer 2 certificate identifies the OS and application executing within the 4758 including information such as program name, developer, program hash, and the application's public key.

Layers 0, 1, and 2 form the TPM and trusted root. The secure co-processor can use its endorsement key in outbound authentications for itself as well as applications running inside it. The TPM generates keys for applications to use in external

[2]Short of physically attacking the memory.

sealed storage and secure channels, and also guarantees these keys never leave the secure co-processor (or if they do, they are encrypted with a key known only to the TPM).

The processing capabilities of the 4758 are significantly less than on integrated trustworthy systems, but the tamper resistance is much stronger, albeit at a higher cost. For these reasons, the 4758 is unlikely to see widespread deployment in commercial DRM installations. However, the 4758 has proved useful in various high assurance applications, and could likewise be used to obtain strong privacy guarantees in PIC systems.

VI. Software Agents

A trustworthy system alone is not sufficient to ensure the protection of personal information. A trustworthy system provides a root of trust from which a trust boundary is extended. In this section, we describe *software agents*, the building block for creating a PIC system built on a trustworthy system.

A. Description

A software agent is a digital representative of an individual. It contains a small amount of verifiable code which it uses to carry out the individual's wishes regarding her personal information and will run only on platforms which the individual specifies. Agents must be able to perform the following functions:

1) Store and report personal data
   Storing and reporting personal data is how the agent represents the individual. The individual should be able to specify fine-grained access control policies concerning this data.
2) Verify platform
   Because the agent can only operate safely on a trustworthy platform, it must be able to verify that it is running on such a platform and refuse to run in any other situation.
3) Open secure channels
   In order to communicate securely with the individual and other entities, the agent must be able to perform authentication and encryption during any communication.
4) Add or delete authorized entities
   The individual must be able to grant or revoke access to her personal data.
5) Log information
   The agent must be able to log any actions that it takes in order to provide a record for the individual to examine.
6) Report information
   The agent must be able to report any action to the individual before it takes that action. The agent must also be able to report the contents of its log.

Figure 1 shows the necessary components of an agent which can perform the functions above. Certain portions of an agent need to be created or initialized by the individual. The code for the agent and a small piece of data which verifies that the code will operate safely on any host must be in the agent. This part of the agent will most likely not change after it is created. Some initial personal data must also be installed on the agent

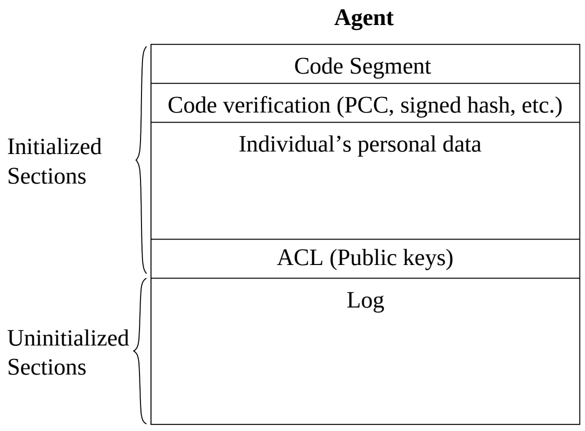

Fig. 1. Agent Composition

as well as an access control policy governing the use of that data. The policy will specify which principals have access by including those principals' public keys. This section can take the form of an access control list (ACL) or any data needed to enforce a capability system[3]. The content of the personal data and access control sections may change during the lifetime of the agent. The log stored here records all of the actions taken by the agent. This log will take the form of a traditional transaction log. The individual need not initialize the log.

An agent with these components can achieve the requirements above. The code verifies the platform on which the agent is running through outbound authentication. It opens secure channels through standard cryptographic mechanisms using keys in the access control policy to identify principals. By using sealed storage, the code can store and retrieve personal data, the access control policy, and the log. The interface to access this data is exported by the code segment.

How the agent achieves the goal of reporting to the individual is not directly solved by trustworthy computing. If the individual wished to receive a log of all the actions taken by the agent, she could simply connect to the agent and request the log over a secure channel. However this approach requires that the individual actively seek out the agent to gather the log. We would prefer that the agent reports and only reports when a new event has occurred. Thus the agent must be able to locate the individual to make its report. We cannot assume that an individual will always be available on a network to receive reports or even that when she is online that she is available at the same network location. The obvious solution is to send an email to the individual. However this solution has the drawback that the agent is then linked to the individual. If an untrusted piece of the system has interacted with the agent and observes the email back to the individual, it can link its interaction with the individual's email address. The problem is to make the receiver of a message unknown to observers. This problem is beyond the scope of this work, but has been treated before[17], [18].

[3]An ACL-based system has the advantage that the individual can revoke access easily.

B. *Agent creation*

The code used in agents should be generic. Because the functionality of the agents is the same, large numbers of individuals should be able to use the same code[4]. What differs between agents should typically only be in the personal data, public keys, and log sections. Of these sections, the only ones explicitly specified by the individual are the personal data and public keys section. Thus the code for agents can be created and verified by vendors once. This code could then be bundled with the appropriate personal data and public keys by a simple program to tailor the agent to a specific individual.

C. *Agent Installation*

Before an individual can use an agent-based PIC-enabled system, she must install an agent on that system to represent her. Installation of the agent requires that she gain assurance about the system and that the system owner gain assurance about her agent. The individual gains assurance about the platform she wants to use through outbound authentication. Typically she will require not only that a valid TPM and trusted root exist, but also certain software is running on the platform. For example, the individual will usually require that there is some way for her agent to report information to her. Once the individual verifies that the system is satisfactory, she will install keys on the agent to enable it to interact with the software on the system. The actual agent can now be transmitted to the system.

At this point, the system owner will want assurance that the individual's agent will be well-behaved on the system. This problem is essentially one of mobile code security. Several methods of assurance could be used in this situation:

1) Formal verification methods such as Proof-Carrying Code[19]
   While formal verification is often expensive or non-scalable, the agent's small size could make this method feasible.
2) Runtime methods such as sandboxing
3) Signed code
   Code for these agents is likely to be very similar for different individuals. This code could be verified and a cryptographic hash of most popular agents could be well known and signed by the verifier.

After both the system and individual are satisfied that the installing the agent will be safe, the system accepts and installs the agent. When the agent is installed, it is placed in SEP storage and the index of the agent is sent to the individual to allow her to access it at a later time. Note that this index does not link the individual to an agent; it only links an agent to its location on the system. In other words, the SEP index of the agent need not be a secret. Whenever any entity, including the individual, wishes to communicate with her agent, that entity need only to ask the system to invoke the agent from SEP storage using this index. Access control is done on the agent after it is running securely. Moreover, the individual can give the SEP index of the agent to any entity whom she wishes and it will not link her to agent. The entity still must go through the agent's access control mechanisms.

In summary, the agent installation process is as follows:

1) Individual connects to a system and verifies it through outbound authentication
2) Individual installs software keys in agent and sends it to the system
3) System verifies the agent
4) System installs the agent in SEP storage and sends the SEP index to the individual

VII. Example Scenarios

In this section, we describe scenarios in which software agents and trustworthy computing platforms can be used to build PIC systems.

A. *Privacy Enhancing Reputation Logs (PERL's)*

A reputation log is a public record containing information about individual's reputation. Examples include credit records and buyer/seller reputations on web services such as Ebay and Amazon. In many reputation logs found today, read and write access is relatively unrestricted: the main benefits are realized largely because the record is public.

Making a reputation log public raises significant privacy concerns. Although access to credit records is limited to those with a justifiable reason, it is not difficult for a determined adversary to gain access to an individual's report and violate her privacy. A more serious problem is preventing unauthorized updates to a reputation log. For a reputation log under a single jurisdiction (such as purchases on the Ebay), it relatively easy for an trusted third party under that jurisdiction (Ebay, in this example) to enforce write access by relating log updates to valid transactions. However, in the more general case (credit reports), it is more difficult to associate log updates to transactions that involve the authentic individual.

This is main challenge of identity theft: individuals have to trust credit reporting agencies that queries and updates to their credit records are indeed valid. There is little economic incentive for a credit agency to put much effort in doing this. The majority of their revenue comes not from individuals, but rather from organizations querying and reporting on those individuals.

Our solution is to use trustworthy computing to create Privacy Enhancing Reputation Logs (PERL's) to enable individuals to directly enforce read and write access control to their own reputation log. However, this solution seems contradictory: a reputation log is meant to public. We justify allowing individuals to enforce read access control by assuming that if an individual denies an organization access to her reputation log, then she will be denied some service in return. This assumption is realistic. Credit records are largely used in part of a decision to grant a loan, issue a credit card, accept a new tenant, etc. If an individual denies access to her credit record then she will likely be denied the loan, credit card, or housing. Justifying giving an individual write access control over her

[4]As an optimization, the code for agents may be shared amongst many agents on the same platform. If there are fewer unique versions of agent code than individuals' agents, then sharing may yield space and performance improvements.

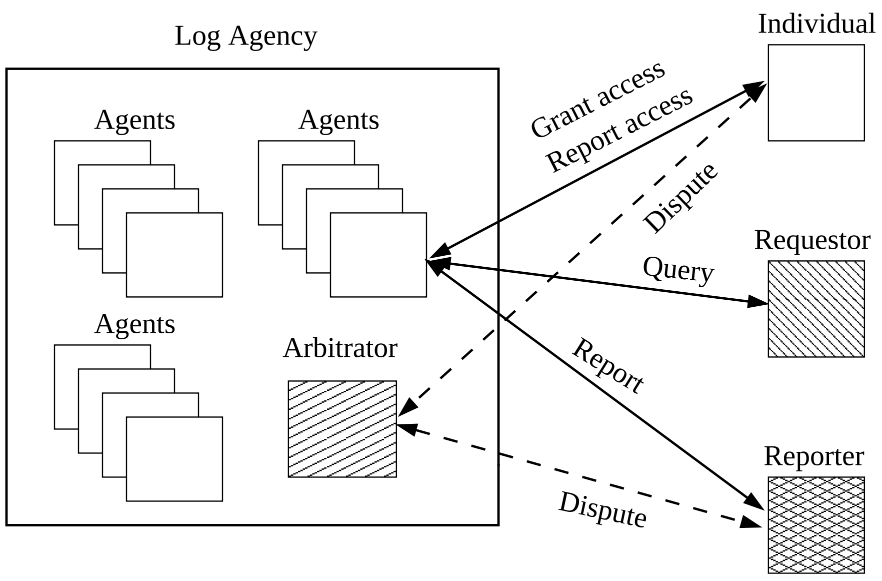

Fig. 2. Reputation Log Architecture

own reputation log is not as direct, and we defer discussion until later.

1) Requirements: In describing PERL's we refer to six principals:

1) The *individual*: The individual is the one about whom the reputation log is maintained.
2) The *agent*: The agent represents the individual on the system on which reputation logs are stored and accessed.
3) A *requester*: An organization or individual requesting reputation information on the individual.
4) A *reporter*: An organization or individual reporting reputation information on the individual.
5) The *arbitrator*: The arbitrator is responsible for resolving disputes between requesters/reporters and the individual.
6) The *log agency*: The system on which individuals' reputation logs are stored and accessed[5]. The log agency hosts the arbitrator as well.

The security requirements for a PERL are as follows:

1) Only parties that have engaged in a transaction with an individual are permitted to write to that individual's PERL. The only exception to this is the trusted arbitrator, who can write to any individual's PERL.
2) Read access control to an individual's PERL is specified by the individual. The only exception is the trusted arbitrator, who can read any individual's PERL.
3) When a third party receives a copy of an individual's PERL, that party should be able to gain assurance of its correctness. Likewise, when an authorized party makes an update to an individual's PERL, that party should be able to gain assurance of the integrity of the update.
4) No deletion is allowed of log records is permitted by the individual. In case of a dispute over an entry in a reputation record, the arbitrator posts a correction entry after investigation.
5) An access history is kept for all read accesses to an individual's reputation record, including reads by the trusted arbitrator.

[5]A naive implementation of PERL's would simply allow individuals to maintain their own reputation log on their own machine. This approach has many problems, the most obvious being that it eliminates one of the main benefits of reputation logs: centralization.

2) Implementation: We now describe an architecture of PERL's based on software agents. Reputation services benefit the most when reputation records are centralized at one highly available reputation agency. Centralized reputation agencies help simplify the task of reporting and receiving reputation records on many decentralized individuals. We adopt this approach in our design as well. Reputation logs on all individuals are stored at a single site, which we call the *log agency*. The log agency needs to be relatively trustworthy with respect to both requesters/reporters and individuals. The log agency is trusted to be highly available and not cause denial of service by maliciously deleting data or denying network access.

The log agency has three primary responsibilities: 1) provide a trustworthy computing platform on which an individual can control access to her PERL by third parties, 2) give third parties requesting and reporting information on individuals assurance about the integrity of that information, and 3) resolve disputes between individuals and the parties with which they interact.

An individual controls access to its PERL by installing an software agent on the log agency's system to represent that individual. A PERL consists of identifying information and reputation log records about that individual. PERL agent code is intended to be fairly simple and re-usable by all individuals. Regardless of who writes the PERL agent, its source code should be open and easily verifiable by any interested party.

To install a PERL agent at a log agency, an individual first prepares the agent with identifying information about the individual and the individual's public signature key. Possible information includes name, address, telephone number, email address, social security number, date of birth, and former residences. The individual first uses outbound authentication to validate the integrity of the log agency's system and then creates a secure execution path on the system for the agent. The trusted root on the log agency returns a SEP index to individual to refer to her agent in the future.

When an individual's agent is first installed at the log agency, the arbitrator generates a random bitstring called the *integrity base* (IB) and installs it in the agent. The log agency provides assurance to third parties requesting and reporting information on individual that the requested and reported information is accurately represented in the individual's PERL by maintaining a *integrity record* (IR) for each agent. The agent's integrity is initialized to be the integrity base. When a new reputation record R_n about an individual is added to the log, the integrity record is updated in the following way:

$$IR_n = H(IR_{n-1}, R_n) \tag{13}$$

where H is a cryptographically secure hash function and $IR_0 = IB$.

By publishing the most recent integrity record and the integrity base for a PERL agent, the log agency provides assurance for both requesters and reporters. When a requester receives a copy of an individual's reputation log from her PERL agent, the requester can easily check the validity of the log by iterating through the hash chain of log records and verifying the final result matches the posted integrity record. Likewise, when a reporter inserts a new reputation record for an individual, the PERL agent provides the log agency with the updated integrity record.

Verifying the new integrity record using only the previous integrity records and its submitted update is not enough enough for a reporter to gain assurance that its update is correctly represented in the PERL. The PERL agent might simply report the correct new integrity record and throw away the update. This will be detected at the next read request and then could be corrected by the arbitrator. Alternatively, the arbitrator could read the latest update from the PERL after the PERL agent has reported new integrity record to validate the result. However, PERL Agents are intended to be uniform, simple, and verifiable, and it is unlikely they would be able to maliciously cheat in such ways. It is not hard, for example, to verify that a PERL never deletes records and always writes them in a monotonically increasing sequence.

When an individual initiates a transaction with a party, it is likely that party will require a copy of the individual's PERL before continuing. A individual can grant the party a read capability identifying the party and its public key, the duration of access, and the SEP index of the individual's agent all signed with her private signature key. The party can then contact the individual's PERL agent at the log agency, authenticate itself to it, and present the capability to receive a copy of the reputation log.

A more delicate issue is granting write access. The justification of allowing an individual to control write access to her own reputation log is to allow her bind updates with valid transactions. This capability is a major step towards empowering individuals to detect theft of their identity. This can be accomplished by granting write capabilities only to parties with which the individual has engaged in a transaction.

The key problem is when to grant the write capability. From the perspective of the party, it is desirable to get the write capability before the start of the transaction. This approach has significant risks for the individual when the party is malicious. A party may initiate a transaction with the individual only to gain read and write privileges and then abort and disappear. But delaying the granting of write access until after the transaction has finished has risks for the party. The individual might behave poorly and then refuse to grant the ability to report on her. Also, how are long running transactions such as an individual's ongoing relationship with a credit card company handled where the company would likely require monthly updates to her log?

We favor delaying the granting of write access as long as possible. Cheating individuals can be handled by reporting the cheating party to the arbitrator along with evidence of the transaction and misbehavior. The arbitrator can then update the PERL with the appropriate record, or simply just insert a "misbehavior" record to serve as a warning to future requesters.

A PERL agent uses sealed storage to record all accesses and updates to the PERL. Occasionally, the PERL agent transfers the access log and new updates back to the individual for review.

Finally, the arbitrator running at the log agency is trusted to be fair and well-behaved. The primary responsibility of the arbitrator is to resolve disputes between individuals and third parties, and to perform this job effectively it requires read and write access to all PERL's on the system. To detect abuses of its power, all actions taken by the arbitrator on an individual's PERL are carefully logged and reported back to the individual.

3) Summary: PERL agents can be effective in controlling access to an individual's reputation log. They not only help individuals control the distribution of personal information contained in PERLS, but also enable individuals to correlate updates to the log with valid, authorized transactions. The major challenge in this application is the choice of when to grant write access to one's log, and how to handle cheating requesters/reporters and individuals.

B. Ecommerce

In this section we will consider an architecture for managing privacy in Ecommerce scenarios.

1) Assumptions: The scenarios that we are considering have the following properties:

1) The individual whose data we wish to protect is a potential customer of an retailer who sells goods or services on the web.
2) The goods or services sold by the retailer require delivery in the physical world to the customer.
3) The retailer may have many customers and is interested in being well known.
4) The customer should be able to access customer service from the retailer.

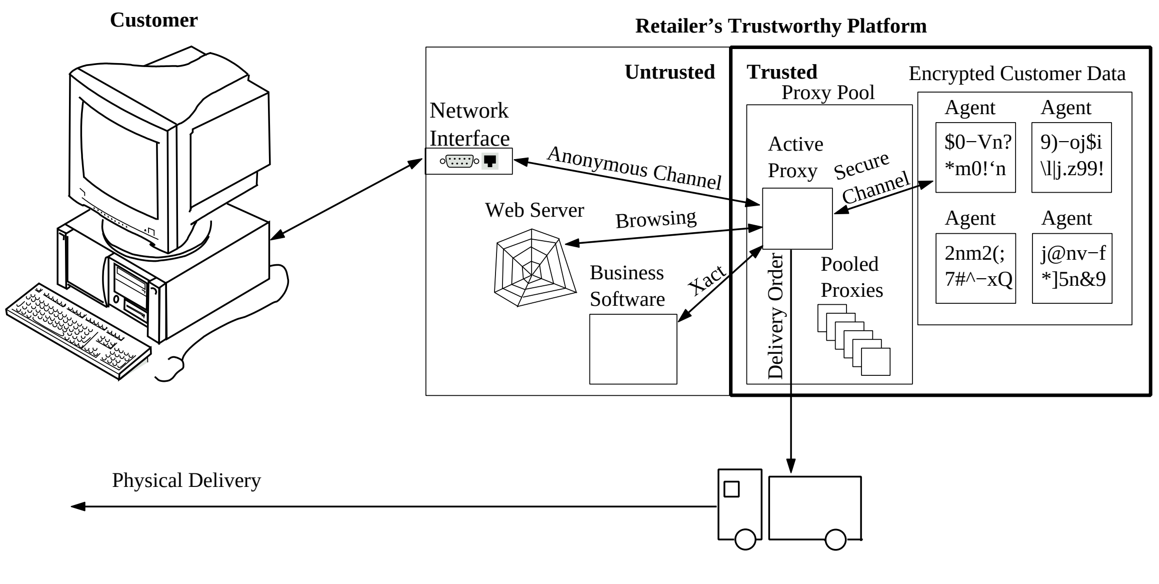

Fig. 3. Ecommerce Architecture

Examples of these situations include ordering a book, flowers, or any other goods or services online.

There are some consequences of these properties. First, the customer will likely search through an online catalog to find the product or service they would like to purchase. Next, because the purchase must be delivered physically, the name and address of the customer must be be revealed to the retailer at some time and likely be printed on an address label. Finally, because the retailer is interested in being well known, it has no reason to hide its own identity. In fact, the retailer is probably interested in having a positive public image.

2) Goals: Our goals in these scenarios are the following:

1) Let the customer control the records of what she looked at in the online catalog.
2) Allow the customer to purchase an item or service and deliver her payment to the retailer while controlling data about the purchase and payment method.
3) Let the customer control all information relevant to delivery of the purchase for as long as possible.
4) Require the retailer to reliably authenticate itself to the customer.
5) Enable customer service representatives to access portions of a customer's data (during authentic, authorized interactions with the customer) while allowing the customer to maintain the same level of control over her data as if she did not require customer service.

3) Architecture: The architecture that we will describe to meet these goals will use the concept of *proxy agents*. A customer will first install her agent on the retailer's trustworthy system, but after installation, all interactions with her agent and the website will be proxied. The only entity which has direct access to the individual's agent will be the proxy. We will see in this section that this approach will allow the individual to meet many of the goals above.

The customer must gain assurance that the retailer is running a trustworthy platform and that it is running a software architecture as shown in Figure 3. Just as certificate authorities are stored in browsers, so too could the certificates for trustworthy, privacy-respecting, proxy software be stored. The retailer merely needs to publish which piece of software it uses. The customer can gain assurance that the retailer's platform conforms to this specification via outbound authentication.

The proxies only have two functions: to authenticate themselves to the customer and her agent in establishing secure channels (most likely with SSL) and to forward requests and replies (between the user, her agent, the webserver software, the preferences software, and the retailer's business software). Because they are relatively small, the proxies could be formally verified to be trustworthy. A proxy has access to agents that customers have installed on the server[6].

In order for the customer to conduct any business with an online retailer, she must connect to that retailer's site[7]. If she has not shopped with this retailer before, the customer must install her agent on the retailer's server. Her agent will contain all personal data that she will use when dealing with the retailer. When the customer connects again, she must ask the proxy to invoke her agent from SEP storage. The proxy can then allow the customer to authenticate herself to that agent.

Once the customer has connected and authenticated herself to her agent, she may begin browsing the store catalog. What she looks at on the retailer's website need not be known to

[6]There could potentially be a pool of such proxies, all of which have access to customer agents. Multiple proxies could improve scalability and performance. In fact, multiple proxies could also improve anonymity – if a customer was not connected through the same proxy consistently, tracking her behavior on the site would be impossible to tie to the behavior of a proxy.

[7]Simply making this connection could reveal information about the individual's interests, needs, or other personal information. This connection could be recorded by the retailer or even an interested third party. We will not offer a way to prevent this information from being recorded with trustworthy computing, but there are existing solutions available which address this problem. In order to preclude anyone from recording the customer's browsing habits, she can use an "anonymizing" service [20], [21] or an anonymous routing protocol [22], [23], [24].

the retailer. All of her website requests go through the proxy. The proxy may remove any information which could identity the user. At that point, the web server software would only see that some sequence of requests are coming from a single proxy in a single session, but it could not link those requests to the individual.

Even though the website cannot associate a history with an individual, the agent can. Information about the customer's browsing history can be stored on the agent by the proxy whenever the customer browses on the retailer's server. This history could then be used to tailor the customer's experience on the website (e.g. suggesting products she might want to buy, showing recently viewed items, etc.), but remain unknown to the retailer if the customer so desired. Without this information, a retailer would have a more difficult time making unsolicited advertisements to customers.

At some point, the agent could forward the customer's (anonymized) history to a *preferences server* (not shown in Figure 3). That server could then compile a preferences profile for the user and pass it in a reply to the agent. This profile would be stored with the rest of the customer's personal data. It could then be passed by the proxy to the webserver during browsing in order to customize the webpage. The preferences profile could either be stored directly on the agent or on the preferences server and accessed via a cookies stored on the agent. Both schemes allow anonymous profiles because all such profiles were generated from anonymous histories delivered via the proxy. Many other methods of building preference profiles (forms, surveys, etc.) could be implemented in a similar way.

Once the customer has selected the items or services that she wishes to purchase, she must submit certain pieces of information to the retailer:

1) The names of the items or services.
2) Relevant delivery information (most likely her name and address).
3) Payment information.

We consider all of this data to be the customer's private data and we would like to allow her control over this information. This information would be transmitted over the secure channel that the customer has with her agent. Her agent would then begin a transaction with the retailer's business software (RBS) to make her purchase:

1) Establish a secure channel between the agent and RBS via the proxy[8]
2) Begin transaction
3) Agent requests allocation of items
4) RBS computes a signed invoice of the available items and transmits to agent
5) Agent consults with customer and if she has decided to make the purchase, agent transmits payment information
6) RBS sends a signed receipt for the payment
7) Agent sends signed receipt and invoice to RBS
8) RBS replies that it has accepted to the order with a signed, paid invoice, or that it has rejected the order and aborted the transaction
9) Commit transaction
10) The agent modifies its ACL to allow the RBS to read the customer's delivery information
11) The agent then sends its SEP index which allows the RBS to retrieve delivery information in the future.

This transaction allows the customer to check the availability of items she wishes to purchase before transmitting her payment information in step 5. Note that this payment does not become associated with a particular invoice until step 7. The items that the customer wants to purchase will not be associated with her in anyway until payment and availability have both been confirmed. The RBS may however decide to abort the transaction in step 8. In this case, payment has been associated with an invoice, but the order was not completed. If the customer decides that this situation threatens her privacy, she may use alternative methods of payment which do not carry identifying information[9]

When the retailer is ready to ship the items to the customer, the RBS will use the SEP index it received in step 11. Determining whether the RBS accessed the delivery information at the appropriate time (i.e. when items where actually available and about to be shipped) is outside of the capabilities of a trustworthy computing system. However, the customer can require her agent to report to her and/or log any such access before divulging any information.

The customer may need to contact customer service at some point after the purchase has been completed. If her agent has not yet received a request to access the delivery information, she may make an inquiry through her agent electronically using the signed, paid receipt that she received in step 8 above. Any communication that the retailer has with the customer will not include delivery information, only payment and invoice data. If the customer's agent has received a request for the delivery information[10], then the retailer is able to link delivery, payment, and invoice information. At this point, all the customer's information is known to the retailer, so traditional customer service could be employed.

4) Summary: Trustworthy computing is able to delay the release of personal information until absolutely necessary. At no point are the customer's preferences or browsing history available to the retailer. The customer is also able to get a signed invoice, payment receipt, paid invoice, and record of the retailer's delivery information request.

There are however some problems that are outside of the realm of trustworthy computing. Analog holes are required in this situation. In order to deliver goods physically, the address of an individual must be known. Printing the address label gives away the name and address of an individual and

[8]The proxy holds the public key (or perhaps even the shared key) of the RBS. It may then create a secure channel with the RBS through which it may route traffic to and from the agent. The agent trusts the proxy, so it may be assured that it is communicating only with an entity the proxy believes is the RBS.

[9]Payment schemes such as electronic cash can remove the need to provide identifying information in the payment information such as a credit card number, name, and bill address. Many such schemes have been described [25], [26], [27].

[10]Any request made by the RBS to access the delivery information must be signed by the RBS and verified by the proxy as part of a secure channel. This signature provides a log that the customer may use to prove that such a request was made.

putting the label on a box associates that name with the goods purchased. These analog holes may be expensive to exploit by reinputting the data manually, but there is another vulnerability which is much faster and cheaper to attack. Because the physical interaction devices (printers, monitors, possibly warehouse robots, etc.) require I/O, it may be possible to snoop sensitive data passing from our trustworthy server to these devices. These digital holes could potentially be closed by requiring all such devices be trustworthy, but this requirement is probably very difficult to satisfy. Moreover, the analog holes inherent in delivery would persist.

VIII. Challenges

While agents are able to offer many advantages over legislation and policy approaches, there are still many challenges to face in PIC systems:

- Incentives for organizations to run PIC systems
  While PIC systems offer many technical advantages, they are only useful if organizations decide to deploy them. Why they would use them is not obvious at the moment because there is much to gain from collecting and distributing or selling individual's information. Some potential incentives for organizations might be new laws mandating PIC systems or customer demands.
- Interaction with current laws
  While we may want to provide the most privacy protection to individuals as is technically possible, this may be unlawful in some cases. Certain countries require retailers to record certain transactions. In cases where personal information is legally required to be revealed, encoding the law into a PIC system is necessary.
- Availability and acceptance of trustworthy computing
  Trustworthy computing platforms are not commercially available as of this writing and not expected to be available for some time. Until such a system is available, empirically testing a PIC system is not possible. Even when trustworthy computing platforms become more widely available, their acceptance is not assured. There has been much controversy over the use of trustworthy systems[28].
- Architecture and systems issues
  Because we cannot empirically test a PIC system at this time, we cannot judge the scalability, performance, and other architectural issues that often arise during actual implementation of systems.

IX. Conclusion

Trustworthy computing is a promising technology for protecting personal information on organizations' computers. The examples that we have given show PIC architectures can be built upon trustworthy computing platforms and offer many advantages over current policy and legislation based approaches. When trustworthy hardware becomes more widely available, PIC systems can easily be installed. However, incentives are necessary for the widespread deployment of PIC systems. Consumer demands and legislation may create this incentive, leading to a more privacy-friendly digital world.

References

[1] D. McCullagh, "Call It Super Bowl Face Scan I," *Wired*, February 2001. [Online]. Available: http://www.wired.com/news/politics/0,1283,41571,00.html

[2] [Online]. Available: http://www.darpa.mil/iao/TIASystems.htm

[3] J. Zaslow, "If TiVo Thinks You Are Gay, Here's How to Set It Straight," *The Wall Street Journal*, 2002. [Online]. Available: http://online.wsj.com/article_email/0,,SB1038261936872356908,00.html

[4] E. Guillermo, "Un-american on a stairmaster: The FBI's house calls." [Online]. Available: http://www.commondreams.org/views01/1218-07.htm

[5] C. Dreher, "Big Brother is watching you read." [Online]. Available: http://www.salon.com/books/feature/2002/02/13/bookstores/print.html

[6] J. S. Stratford and J. Stratford, "Data Protection and Privacy in the United States and Europe," May 1998. [Online]. Available: http://www.iassistdata.org/publications/iq/iq22/iqvol223stratford.pdf

[7] S. Andrews, "Privacy and Human Rights: An International Survey of Privacy Laws and Developments," 2002. [Online]. Available: http://www.privacyinternational.org/survey/phr2002/

[8] T. C. Greene, "Ms digital rights management scheme cracked," *The Register*, October 2001. [Online]. Available: http://www.theregister.co.uk/content/4/22354.html

[9] "CloneCD." [Online]. Available: http://www.elby.ch/english/products/clone_cd/index.html

[10] "Efforts to stop music piracy 'pointless'," *BBC News*, November 2002. [Online]. Available: http://news.bbc.co.uk/1/hi/technology/2502399.stm

[11] P. Biddle, P. England, M. Peinado, and B. Willman, "The Darknet and the Future of Content Distribution," November 2002. [Online]. Available: http://crypto.stanford.edu/DRM2002/darknet5.doc

[12] "Trusted Computing Platform Alliance." [Online]. Available: http://www.trustedcomputing.org/

[13] "Microsoft "Palladium": A Business Overview." [Online]. Available: http://www.microsoft.com/presspass/features/2002/jul02/0724palladiumwp.asp

[14] S. Smith, "Outbound authentication for programmable secure coprocessors," in *7th European Symposium on Research in Computer Science*, October 2002.

[15] J. Dyer, M. Lindemann, R. Perez, R. Sailer, S. Smith, L. van Doorn, and S. Weingart, "Building the IBM 4758 secure coprocessor," *IEEE Computer*, vol. 34, no. 10, pp. 57–66, October 2001.

[16] S. Smith and S. Weingart, "Building a high performance, programmable secure co-processor," *Computer Networks*, vol. 31, no. 4, pp. 831–860, April 1999.

[17] A. Engelfriet, "Mixmaster remailers." [Online]. Available: http://www.stack.nl/ galactus/remailers/index-mix.html

[18] I. Goldberg, D. Wagner, and E. Brewer, "Privacy-enhancing technologies for the internet," in *Proc. of 42nd IEEE Spring COMPCON*. IEEE Computer Society Press, Feb. 1997. [Online]. Available: http://citeseer.nj.nec.com/54687.html

[19] G. C. Necula, "Proof-carrying code," in *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Langauges (POPL '97)*, Paris, Jan. 1997, pp. 106–119. [Online]. Available: http://citeseer.nj.nec.com/50371.html

[20] [Online]. Available: http://www.anonymizer.com

[21] "The Lucent Personalized Web Assistant." [Online]. Available: http://www.bell-labs.com/project/lpwa/proxy_index.html

[22] "Onion routing." [Online]. Available: http://www.onion-router.net/

[23] M. K. Reiter and A. D. Rubin, "Crowds: anonymity for Web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, pp. 66–92, 1998. [Online]. Available: http://citeseer.nj.nec.com/284739.html

[24] M. J. Freedman, "A Peer-to-Peer Anonymizing Network Layer," Master's thesis, MIT, 2002. [Online]. Available: http://citeseer.nj.nec.com/freedman02peertopeer.html

[25] B. Yee and J. D. Tygar, "Secure coprocessors in electronic commerce applications," 1995, pp. 155–170. [Online]. Available: http://citeseer.nj.nec.com/tygar95secure.html

[26] J. Camp, M. Harkavy, J. D. Tygar, and B. Yee, "Anonymous atomic transactions," in *Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, 1996, pp. 123–133. [Online]. Available: http://citeseer.nj.nec.com/camp96anonymous.html

[27] D. Chaum, "Security Without Identification: Transaction Systems to Make Big Brother Obsolete," vol. 28, no. 10, October 1985. [Online]. Available: http://www.chaum.com/articles/Security_Wthout_Identification.htm

[28] R. Anderson, "TCPA / Palladium Frequently Asked Questions." [Online]. Available: http://www.cl.cam.ac.uk/ rja14/tcpa-faq.html