CHAPTER 8

Regulatory Approaches

Engineers in general and computer scientists in particular don't frequently view regulation as an appropriate way to solve technical problems.

There are many reasons that technologists are predisposed to avoiding regulatory solutions. Most have to do with how the regulatory process works in practice:

1. Because laws are passed in a democratic process, actors opposed to the law have a seat at the table when the law is drafted. As a result, many laws are watered down.
2. After a law or regulation is passed, it needs to be enforced. One of the frequent criticisms of the CAN-SPAM law outlawing junk email [CAN03] is that the amount of spam on the Internet significantly increased after the law was passed.[Car04]
3. Laws and regulations are open to misinterpretation. Many engineers and others feel that lawyers can distort a good law beyond recognition in the courts.
4. Policy solutions do not work across legislative boundaries. The CAN-SPAM law, for instance, is not effective against unwanted email that originates outside the United States and advertises foreign goods or services.

Policy solutions can never be 100% effective. Horrible activities like murder and genocide have not been eliminated through the use of policy, even though these activities are illegal.

The hesitancy with which many engineers approach policy solutions may have more to do with the engineers themselves. Few engineers have training in law and policy, for example, making these solutions seem more ad-hoc and less likely to work. Indeed, many of the criticisms of regulatory solutions can be easily applied to technical solutions as well:

1. Because of the need to preserve backwards compatibility, many new engineering solutions are incomplete.
2. After a technology is designed and deployed, the environment continues to change. Thus, solutions that work today may suffer "bit rot" and cease to work in the future.

3. Because it is difficult to anticipate all possibilities in advance, pathological "corner cases" exist in most engineering systems which break the intended solution.

4. Technological solutions are frequently wed to a particular problem domain. For example, an anti-spam solution that runs on Unix may not work properly on the Windows operating system. An anti-spam system that works for spam written in English may not work properly with spam written in Japanese.

Engineering solutions are rarely if ever 100% effective. Horrible security vulnerabilities like buffer overflows and privilege escalation attacks have not been eliminated in production software, even though these techniques for addressing these problems (*e.g.,* type-safe languages and formal privilege specifications) are well understood.

8.1 Patterns for Regulation

Two patterns are proposed for increasing the alignment of usability and security:

- **CREATE A SECURITY LEXICON** (page 340)
  The security lexicon is a standard vocabulary that is used to discuss security issues. To decrease the potential for confusion, the same lexicon should be used among both users and security practitioners.

- **DISCLOSE SIGNIFICANT DEVIATIONS** (page 341)
  When an object (software or physical) is likely to behave in a manner that is significantly different from that which user expects, those differences should be disclosed. Ideally, those differences should both be disclosed before the object is acquired and during the object's use.

Evidence for the appeal of these patterns can be found both in the century-long history of regulating food and drugs in the United States, and in the recent history of regulating disclosing privacy practices of Internet web sites. In both cases, it has been the intent of rules, regulations and standards to first establish a *common vocabulary*, and second to *disclose* ingredients, effects, or practices that the user would not otherwise be able to infer.

There is a strong parallel between 19th Century adulterated food products discussed earlier in Chapter 2 and 21st Century adulterated software. Just as some tonics claimed to do one thing (like sooth a disruptive child) but had a significant hidden function (making the child intoxicated and chemically dependent on an addictive drug), today we have software that claims to do one thing (set the time of your PC) and has a significant but hidden function (displays ads when the computer user visits particular web sites).

There is also a strong parallel between disclosing the information-handling practices of web sites, which by their nature are not obvious to visitors but which can have profound impacts at a later time, to the presence of radio tracking devices and readers in consumer goods and the environment—which, once again, are not obvious but which may have profound impacts.

8.2 The Security Lexicon

In an interview at RSA Security, Debbie Stolper stated that the origin of many usability problems RSA is different are used inconsistently within the company's products to represent similar concepts.[Sto04] The obvious solution for this problem was for RSA to agree upon a security lexicon for use within a family of products. At the time of the interview, use of that lexicon was being expanded to other product families within the company.

The need for consistent language and a unified lexicon has been noted by many in the usability field. (*e.g.,* [Joh00, p.206]) But vocabulary rarely receives a treatment that is deep or systematic: usability professionals simply note that it is important that a single set of terms be used for concepts within the system that is being analyzed.

This section holds that there are deep, understandable and systematic reasons why computer systems tend to employ inconsistent language. By specifically focusing on the need to use clean and standardized language, confusion can be averted.

8.2.1 Confusing security terminology

The idea that verbal ambiguity might lead to usability problems in the field of security is a frequent topic in the hallways of security conferences. Nevertheless, there appears to be no previously published scholarly work exploring this topic.

It is easy to toss stones at the lack of lexical purity within the computing field. Information technology practitioners frequently respond to the ambiguity of human language by coining new words or acronyms for new concepts. Because IT creates so many different artifacts and the underlying technology changes so fast, a proliferation of terms appears inevitable.

Many factors, including the lack of standardization, flexibility of implementation, and user customization, result in both the same underlying concept being described by different terms, and in the same term being used to describe different concepts. For example, Douglas Engelbart coined the term *mouse* in 1964 to describe a particular kind of X/Y pointing device [Eng67], but today the term is used somewhat generically (if incorrectly) to describe trackballs and touch pads as well. Such misuse is not merely colloquialism: the MacOS 10.3 System Preferences Panel uses the label "Keyboard & Mouse" as the entry point for the control panel that controls the system trackpad (Figure 8-1).

Security has been especially susceptible to vocabulary creep because of the rapid innovation rate. In a fast-moving field, one way to stake a claim is create a terminology and try to get people to use it. It is also appropriate to change terminology to denote the lack of compatibility. Today Internet Explorer version 6 (SP2) has support for SSL 2.0, SSL 3.0, and TLS 1.0. It's good that the distinctions are clear to security professionals and developers, but they probably don't need to be made visible to the user.[1]

[1]Nor does it make sense for Internet Explorer 6 to give the user the ability to individually enable or disable each of these protocols: all should be enabled, for if a serious security vulnerability is found with any of them, Microsoft would almost certainly distribute a patch using Windows update. The tendency of Microsoft and other companies to expose such policy decisions to the end users—who are generally not equipped with the necessary knowledge to make a decision—rather than making the correct decision on behalf of the user is discussed in Section 9.4.

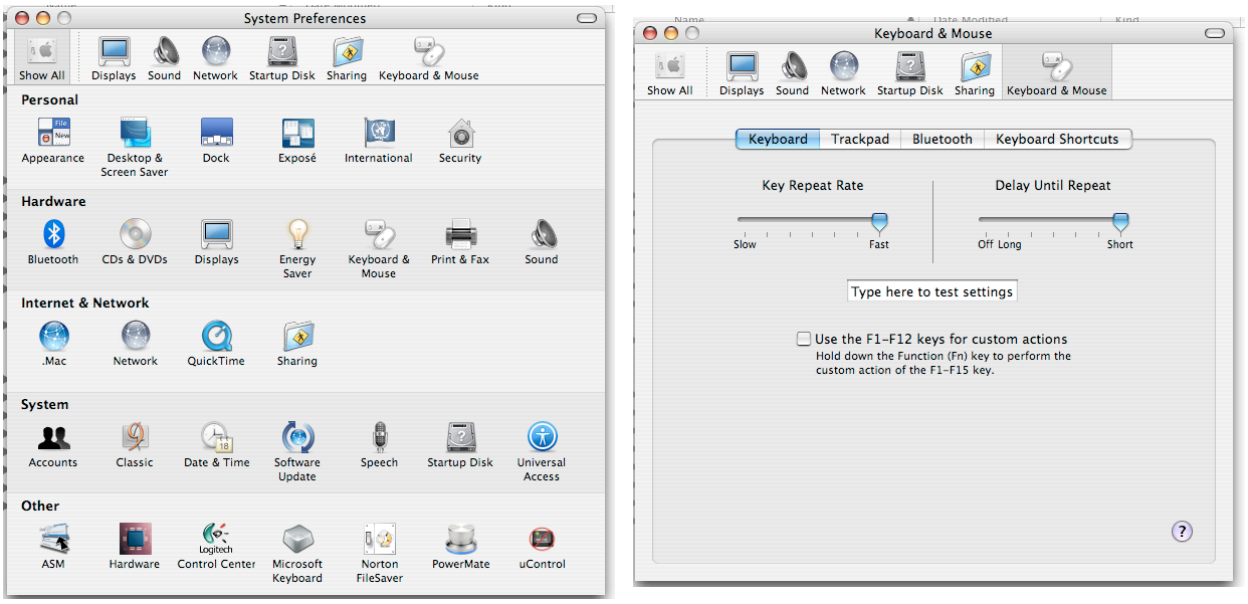

Figure 8-1: The Apple MacOS 10.3 System Preferences panel incorrectly uses the term "mouse" to describe the trackpad when the operating system is running on a PowerBook G4.

New terminology is also sometimes introduced in an attempt to make old work look new. The terminology problem is further complicated by the fact that it is easy to re-invent computer techniques, making it quite possible that two terms will be independently adopted to describe what is more-or-less the same underlying technology.

Keys, Certificates and Digital IDs

Sasse has argued that, more than other specialties, security practitioners are guilty of taking terms that have "real-world meaning" to describe security concepts that are similar but significantly different. Her two primary examples are the words "key" and "signature" which are used to describe strings of bytes with particular properties, rather than a metal object used for opening doors (in the first case) and a sample of handwriting on a piece of paper (in the second case).

"This is a fundamentally broken model," Sasse argues. "It would be much better to change your terms and build new models from scratch."[Sas04a]

Equally broken is the inability to standardize on a lexicon for even the most basic computer security concepts.

For example, what is the proper term for the mathematical complement of a "public key:" a "secret key" or a "private key?" Both terms have been used, although the community seems to be settling on the notion that "private key" is the complement while "secret key" is a term to be used only with symmetric cryptography. But these conventions are neither standardized nor rigorously followed.

For example:

- Cormen, Leiserson and Rivest use the term "secret key:"

> "In a public-key cryptosystem, each participant has both a **public key** and a **secret key**." [CLR90, p.831], [CLRS01, p.881]

- Schneier uses term "private key."

  > "In these systems, the encryption key is often called the **public key**, and the decryption key is often called the **private key**."[2] [Sch96, p.5], [FS03]

- Stallings uses the term "private key:" [Sta03, p.260]

- Whitten's dissertation uses the phrase "private key" *both* to describe symmetric key cryptography and to describe the complement to one's public key. She also uses the phrase "secret key" to describe the complement of the public key:

  > "The user test was run with twelve different participants, all of whom were experienced users of email, and none of whom could describe the difference between public and private key cryptography prior to the test session." [Whi04a, p.18]
  >
  > "In order to complete this task, a participant had to generate a key pair, get the team members' public keys, make their own public key available to the team members, type the (short) secret message into an email, sign the email using their private key, encrypt the email using the five team members' public keys, and send the result." [Whi04a, p.16]
  >
  > "Objects: Key pairs, each consisting of a secret key and a public key." [Whi04a, p.42]

- Salomaa largely avoids the questions of "private" vs. "secret" in his introduction to the RSA algorithm [Sal96, pp.125–128] by referring to combination of the RSA modulus and encryption exponent as the *public encryption key*, and referring to $p, q, \phi(n)$ and d as the *secret trapdoor*. Although this colorful language parallels both Salomaa's previous chapters and the original *New Directions* paper [DH76], it is likely that users of desktop software would be confused if the term "secret trapdoor" were used to describe the complement of the user's public key.

The purpose of this survey is not to put the question of *private* vs. *secret* up to a popular vote (perhaps weighting each author by their publication count), or to embarrass authors who have used the terminology inconsistently, but instead to show that the vocabulary of public key cryptography is both confusing and ultimately dangerous.

Indeed, the lack of linguistic consistency in our security tools can have real-world security consequences. In *Lessons Learned in Implementing and Deploying Crypto Software*,[Gut02a] Gutmann is particularly critical of both the PKCS #12 standard[RSA99] *and* the use of the word *Certificate* or *Digital ID* to describe a PKCS#12 file.

Originally introduced by Microsoft in 1996 as the PFX (Personal Information Exchange) file type, the PKCS #12 file can contain a public key, a certificate, a password-protected private key, or any combination of those three elements. Because the file can contain private key material, Gutmann asserts that it is improper and even dangerous to refer to a PKCS#12 file as a "certificate" or "Digital ID." To make matters worse, the Windows "Certificate Export Wizard" actually creates PKCS #12 files as output, and by default will export *both* the certificate and its corresponding private key.

[2] Schneier notes "The private key is sometimes also called the secret key, but to avoid confusion with symmetric algorithms, that tag won't be used here."

> "The situation is further confused by some of the accompanying documentation, which refers to the PKCS #12 data as a 'Digital ID' (rather than 'certificate' or 'private key'), with the implementation that it's just a certificate which happens to require a password when exported.
>
> "The practice of mixing public and private keys in this manner, and referring to the process of and making the behavior of the result identical to the behavior of a plain certificate, are akin to pouring weed killer into a fruit juice bottle and storing it on an easily accessible shelf in the kitchen cupboard.[Gut02a, p.4]

To prove his point, Gutmann relates specific cases in which this use of incorrect terminology compromised private keys. In one case, Gutmann was sent the private key and matching certificate necessary to authorize access to third-party financial records in a European country. In another case, a CA distributed a PKCS #12 file containing the CA's root key and certificate to relying parties.

8.2.2 "Trusted" and "non-repudiation"

Similar problems can be found in two other words that litter computer security: "trusted" and "non-repudiation."

In *Ten Risks of PKI: What You're not Being Told about Public Key Infrastructure*, Ellison and Schneier argue that individuals and corporations promoting PKI took several specific words from the jargon of academic cryptography and interjected those words into marketing literature and—eventually—into legislation. The purpose of this linguistic transfer was to improve acceptance of PKI technology by making PKI-based systems appear to be more secure than they actually are.[ES00]

Ellison and Schneier are particularly critical of the words *Trusted* and *Non-Repudiation*:

- Trusted, they note, is used in marketing literature to imply that certificates issued by the CA can be relied upon for a particular purpose, while the academic literature uses the term to mean "that [the CA] handles its own private keys well."
- Non-repudiation is used in marketing literature to mean "if your signing key has been certified by an approved CA, then you are responsible for whatever that private key does. It does not matter who was at the computer keyboard or what virus did the signing: you are responsible." But the technical meaning of *non-repudiation*, they argue, is "the digital-signature algorithm is not breakable, so a third party cannot forge your signature."

There is anecdotal evidence that the heavy emphasis on "non-repudiation" by promoters of digital signatures in the 1990s may be responsible for the unwillingness of many organizations to embark on aggressive campaigns to promote the use of digital signatures: these organizations did not want every email message to carry the weight and authority of a signed contract.

This aggressive promotion appears to have backfired.[Bid96] For example, although Utah passed legislation to promote the use of digital signatures[Uta95] and VeriSign initially registered as a CA under that law,[Las97] VeriSign did not renew its registration when it expired. Reportedly, there was no market for such legally binding high-assurance digital signatures.

Ultimately, the use of digital signature technology in e-commerce is going to depend on easily understood and broadly accepted definitions of the terminology present in the interface, the standards, and the legislation. If those definitions go too far, or if there is a disconnect between the legal assurances that are offered and those that the users actually need, progress will not be made.

8.2.3 "Delete," "erase," "purge," "clear," and "wipe"

Today many different words are used for the act of expunging information from a computer system. "Delete" and "erase" are two common words that are commonly used as synonyms. But "purge" is another word that's commonly used—is it different? Sometimes the term "clear" is used to indicate that media has been overwritten with a single pass ASCII NUL characters; in other cases, "clear" is used to indicate that the data has been overwritten with several passes according to DoD 5220.22-M[DoD95], but that the standards necessary for sanitizing media of classified information have not been followed.

The confusion over verbs for the act of expunging information mirrors very closely the confusion over what actually happens when a user asks for information to be expunged. When standards are adopted for the removal of information, those standards should similarly specify which words are used to describe the process.

8.2.4 "Digital signatures" vs. " signatures"

The Microsoft Outlook Express 6 program studied for the *Johnny 2* experiment confusingly uses the phrase "digital signature" to describe a cryptographic operation, while it uses the phrase "signature" to indicate the text that is automatically appended to the bottom of every outgoing message. As a result, the program has a check-box that reads "Digitally sign all outgoing messages" (see Figure 8-2) and a second check-box that reads "Add signature to all outgoing messages" (see Figure 8-3). These two commands have very different effects.

8.2.5 Recommendations

This isn't an area that needs more research so much as it needs decisive action. Some organizations organizations have in-house "style books" that are lexicons of which words to use in which situations. Unfortunately, even organizations that have adopted standard terms are inconsistent in their uses.

For example,the term 'Digital ID" has become a standardized term to describe an X.509 certificate used to identify an individual. The term is used by Microsoft consistently in its products and on its web sites. "Digital ID" is a good term that most users in the Johnny 2 user test understood immediately. Unfortunately, VeriSign uses the term term intermittently on its web site to describe both certificates used for code signing and servers. In other places the phrase is not used at all.

Such a security style book should be subject to user testing before it is adopted. The style book could also include images and icons for standard concepts as well, taking into account Whitten's "metaphor tailoring."

The security style book has got to be a joint project of the major software and security vendors so that the same concepts are represented by the same terms in all user-facing programs. Such a style

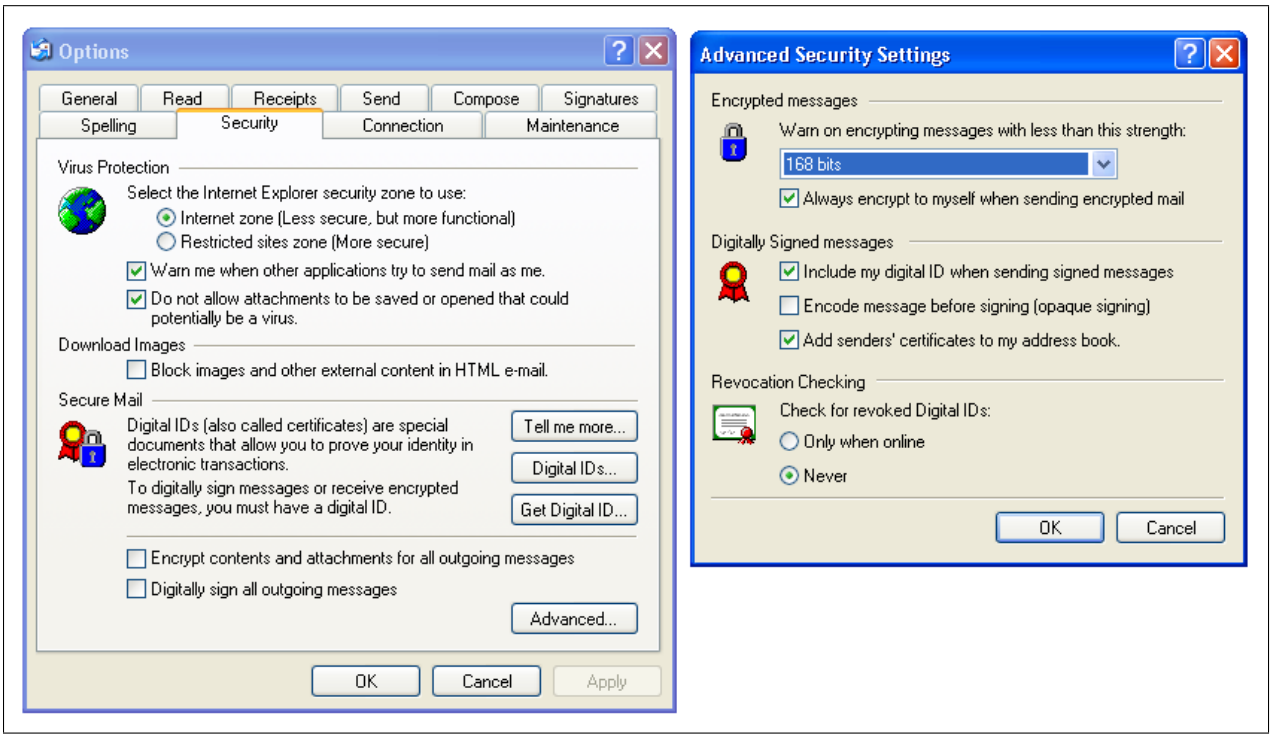

Figure 8-2: The OE6 “Security” options panel controls virus protection, downloaded images, and the S/MIME functionality of the program.

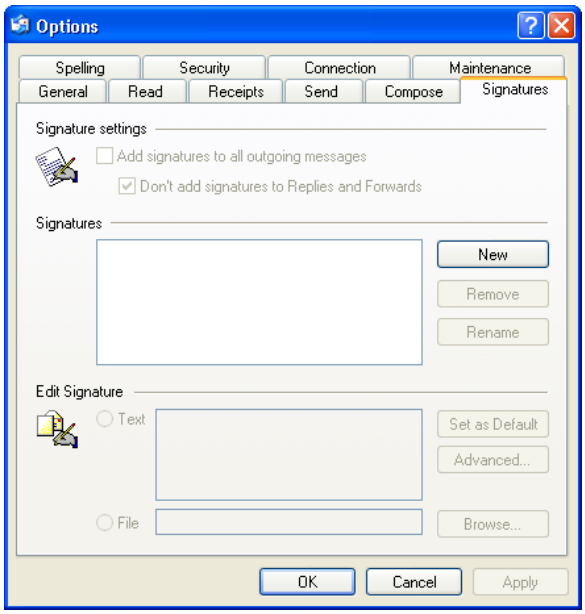

Figure 8-3: The OE6 “Signatures” options panel is not for digital signatures, but for textual signatures which OE6 can be instructed to place at the bottom of every email message.

book could even be created by a standards committee. Once the style book is adopted, companies need to be willing to transition their existing programs to the new nomenclature.

Standardizing on a common security terminology is not a theoretically interesting problem, but it is an important part of making security usable.

8.3 Spyware and the "Pure Software" Proposal

Spyware is the scourge of desktop computing. Yes, computer worms and viruses cause billions of dollars in damage every year. But spyware-programs that covertly monitor the user's actions and report them to a another party are deceptive in ways that most computer users find morally repugnant.

8.3.1 Evidence of the spyware problem

Evidence of the spyware problem is widespread:

- The Internet service provider Earthlink offers a free "Webroot" scan service to its customers. During the first four months of 2004, the company scanned 1.5 million systems and found some kind of remote system monitor or Trojan horse on roughly a third of those systems.[Gra04]

  Reports of surveys such as this are frequently suspect because the most popular spyware detector programs, Ad-Aware and Spybot Search and Destroy, report cookies from web site such as Doubleclick as instances of "spyware" which they then offer to remove. There is a perverse incentive for makers of defense programs to report dangers when none exist, and to over-emphasize the danger of the dangers that they actually find. But the Earthlink survey didn't fall into this trap: the company separately reported the different types of spyware found on the systems. Of those 1.5 million, the company found 257,761 Trojans installed, 245,432 remote system monitoring tools, 7,642,556 adware installations, and 32,700,340 "adware cookies."

- A survey of 329 home computers conducted by technical experts on behalf of the Internet service provider AOL in September and October 2004 found various forms of spyware and adware on 80% of the computers. "About 90% of those whose computers were infected with spyware didn't know about the infections and didn't know what spyware programs are, the survey showed."[Rob04b]

- In August 2003 spyware accounted for 1% of tech support calls to Dell. This number jumped to 12% in early 2004 and 20% in October 2004. Dell concluded that "more than 90 percent of computers in the United States contain some form of spyware," and decided to include an anti-spyware program on all new computers sold.[Ger04]

- Jeffrey Friedberg, Microsoft's director of Windows privacy, told Congress in Spring 2004: "We have evidence that [spyware] is at least partially responsible for approximately half of the application crashes our customers report to us.... It has become a multimillion-dollar support issue."[Sca04]

According to an article in the November 2004 issue of *CSO Magazine*, computer security professionals are now viewing spyware as one of their primary security concerns. [Sca04] Consumers seem

Sharman's No Spyware Commitment

- Kazaa does NOT install or delete software from your computer without your permission.
- Kazaa does NOT contain software that gathers personally identifiable information about you.
- Kazaa and its partners securely process any credit card or transaction information you may give.
- Kazaa does NOT contain software that monitors keyboard strokes.
- Kazaa does NOT deceptively install software that centrally records your personally identifiable Internet usage.
- Kazaa does NOT prevent your efforts to remove Kazaa.

The official, certified versions of Kazaa software - Kazaa and Kazaa Plus - are accessible from Sharman Networks and its authorized publishers through www.kazaa.com.

Kazaa, which is supported by advertising, and Kazaa Plus, which is not advertising supported, do not deliver software – which we refer to as "spyware" – that is installed without your prior consent or that gathers any personally identifiable information without your consent.

Unofficial, fake or hacked versions of the software might include spyware. These are not products of Sharman Networks or its authorized publishers and always infringe the Kazaa copyright. Use caution before downloading and/or installing all software.

Figure 8-4: The "No Spyware Commitment" from Sharman Networks does an excellent job saying what Kazaa does not do, but it says little about what functions are performed by the advertising-supported software including with the free version of the product.[Sha04]

to be more sanguine; according to Dell, the primary consumer complaints about spyware isn't the fact that personal information is potentially compromised, but the argument that spyware makes computers run slower. [Del04b]

One unfortunate problem with these surveys and statistics is that there is no concrete definition for what "spyware" actually is. Programs that record keystrokes and screen contents certainly seem to fit the definition, but what about programs that simply monitor visited web sites and display advertisements from competitors? These programs might seem like "spyware" to people who are running them without their knowledge, but the companies that are distributing the programs—companies like Sharman Networks, makers of Kazaa—insist that these programs are really "adware" and that they generate the revenue that pays for the development of software that is made available for free download. In fact, Sharman actually distributes two versions of Kazaa—one version that is free but "ad supported" and features adware from GAIN Network, and another version that costs $29.95 and includes "No Ads." (See Figures 8-4 and 8-5.)

The computer industry has focused on technical means to control the plague of spyware. Programs such as Ad-Aware[Lav04] will scan a computer for known spyware, tracking cookies, and other items that might compromise the user's privacy. Once identified, the offending items can be quarantined or removed. Firewall programs like ZoneAlarm[Zon04] takes a different approach: they don't stop the spyware from collecting data, but they prevent the programs from transmitting personal information over the Internet.

Figure 8-5: The advertisement for Kazaa v3.0 makes it very clear that the free version of the program comes with adware and the $29.95 version does not. Yet many people who download and run the free version are surprised to discover that their computers run adware as a result.

But why are these programs installed in the first place? Perhaps because many people simply do not read information about software before they download and run it—an observation in line with the findings of the European Heart Network[EHN03] regarding consumers failure to read and process information on food labels discussed in Section 2.6.2.

8.3.2 The "Pure Software" proposal

Part of the spyware solution may be a software labeling standard that discloses specific functionality within software that people find objectionable. As was the case with soothing syrups, the mere requirement of labeling may cause some of the most objectionable software practices to be discontinued. And once there is a simple and well-defined set of functionality that has been identified, consumers could be educated to at least look at these labels and perhaps use the information. Such a labeling requirement could work because most organizations authoring and distributing spyware are not criminals or pirates, but legitimate organizations trying to earn money within the legal economy.

Congress could pass legislation requiring that software distributed in the United States come with product labels that would reveal to consumers specific functions built into the programs. Such legislation would likely have the same kind of pro-consumer results as the Pure Food and Drug Act of 1906—the legislation that is responsible for today's labels on food and drugs.

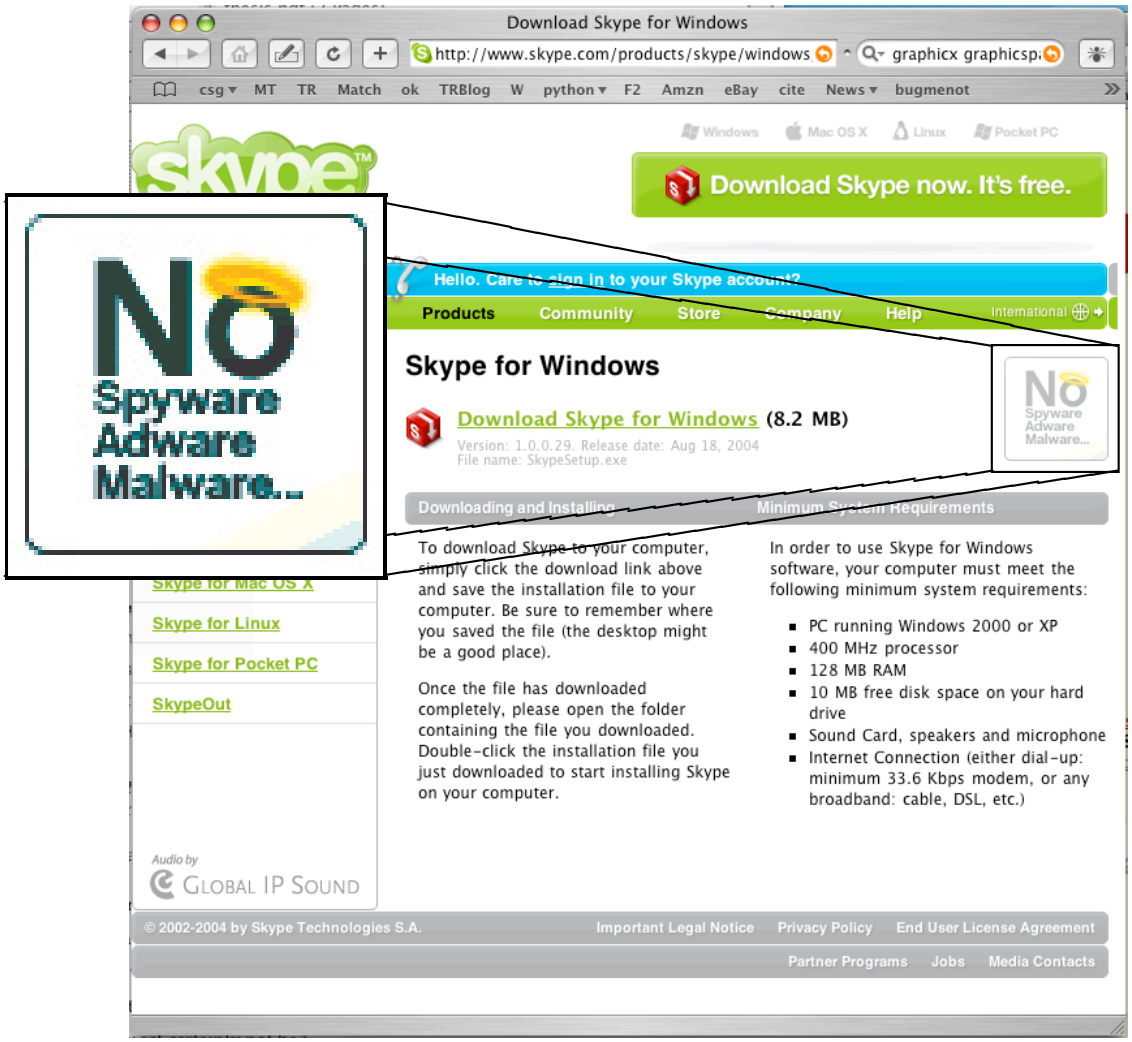

Figure 8-6: Spyware has become so pervasive in popular downloaded software that the makers of Skype need to explicitly state that their software contains "No Spyware, Adware [or] Malware." Sadly, these terms are not defined.

Mandatory software labeling is a good idea because the fundamental problem with spyware is not the data collection itself, but the act of deception. Many of the things that spyware does are done also by non-spyware programs as well. Google's Toolbar for Internet Explorer, for example, has two versions. One version reports back to Google which web site the user is visiting so that the toolbar can display the site's "page rank;" the other version does not report which web site is being viewed, but likewise does not display the site's rank. Google discloses this feature when the user installs the program—forcing the user to decide which version of the toolbar is to be installed. "Please read this carefully," says the Toolbar's license agreement, "it's not the usual yada yada."[Goo04]

Most spyware, on the other hand, goes out of its way to hide its true purpose. The program Precision Time[GAI04] has a heavily publicized function that automatically sets a computer's clock from the atomic clock operated by the U.S. Naval Observatory. But the program also displays pop-up advertisements that are part of the GAIN Network.

What's more, there is increasing evidence that makers of spyware have deliberately designed their programs to be difficult to detect and difficult to remove. [OH04, Lub04]

8.3.3 Crafting the policy

Building on the experience that the US has had with regulating adulterated foods, the first step towards solving the spyware problem would be to enact legislation that would directly address the issue of deceptive software features. There is broad agreement throughout human society that deception is morally impermissible.

There are many ways that software can engage in deception. For example, Luber notes that "small print in the terms and conditions" of a particular application "reveals that [the software] will change the user's home page, install bookmarks leading to adult and/or sexual content, and deactivate browser toolbars."[Lub04] Clearly, it is not sufficient to mandate that information such as this be disclosed: it must be disclosed *prominently*.

The basis of any "Pure Software" regulation would thus need to consist of several elements:

- The regulation should require that all programs running on a computer reveal themselves though the standard means used by the host operating system. On Windows, this would be implemented by forcing running programs to appear in the task bar or in the icon tray.
- The regulations would require that all programs have an "uninstall" feature that actually uninstalls all traces of the program.
- The regulations would specify specific kinds of functions that would have to be explicitly revealed when programs are distributed and run. Programs that implemented these kinds of functions would need to make that functionality clear to users at download, when the software was installed, and when it was run.
- Instead of allowing companies to hide the identified functions in obscurely written legalese buried in click-through license agreements, the regulations would require that the disclosure be made in the form that is easy to understand. Based on the experience with other labeling efforts in Information Technology, it seems that the easiest labels would be simple and distinctive icons that could be clicked on for additional information.
- These icons would be displayed in specific, uniform places. For example, in the Windows operating system such places would include the Task Manager and the Add/Remove control panel could both display the mandated behavior icons alongside the program's application icon.
- Clicking on the icon would bring up further explanatory text—perhaps from a web site maintained by the Federal Trade Commission.

It is important that the number of identified functions be limited: if there are too many, then software installations will consist of a sea of icons and no usability objective will be accomplished. The *Principle of Least Surprise* is an excellent guidepost in deciding upon these functions: they should be functions that *surprise* the user by their existence or when they run.

For example:

- On a computer with a desktop metaphor, for instance, each application's window can be seen as a form of containment: programs that violate this containment behave in a surprising fashion. When a program's window is active, it is expected that this program will be receiving keystrokes; when a program is not active, it is expected that it will not.

- Users are frequently surprised when they discovered that one program—say, a word processor—can read the files created by another program—say, a spreadsheet. If a game that is downloaded from the Internet starts scanning through the user's hard disk searching for text files that contain credit card numbers, this is surprising behavior.

- As Yee discusses, software on a computer has considerable more powers and capabilities than most users suspect.[Yee05b] If a program that is downloaded for the purpose of viewing pornography has the effect of disconnecting the user's computer from one ISP and placing a long-distance telephone call to another ISP—perhaps one that requires an expensive long-distance telephone call—this is surprising behavior.[Fed97]

Below a sample list of eight possible surprising behaviors are proposed in detail, each item accompanied by an illustrative icon.[Gar04a] These icons are intended for illustrative purposes only: the actual government-mandated icons would need to be developed by a team of professionals with expertise in human computer interface, user tested, and put up for public comment. But these icons are useful to convey the general idea and to start a discussion:

Dial: Places a Phone Call
One common spyware scam involves programs that cause a computer to call phone numbers that cost more money than a normal phone call. For example, in 1997 some pornographic web sites distributed a program called `david.exe` that caused the victim's computer to make a long-distance phone call to an Internet service provider in Eastern Europe; the porn company got to keep half of the (exorbitantly high) long distance revenues.[Fed97] Programs that dial the phone are a significant problem in Germany and in other European countries with "caller pays" billing plans. Documenting that the software has code that intended to dial the phone—either autonomously or in response to a user's command—would be a good way to address this problem.

Hook: Runs at Boot
Some programs hook themselves in to the computer's operating system so that they automatically run whenever the computer is rebooted or a user logs in. Other programs don't. Today there is no way to tell except by performing a detailed analysis of the computer's configuration files before and after the program is installed and noting the changes. Any program that installs itself so that it automatically runs would have to display this Hook icon.

Modify: Alters The Computer's Operating System
Some programs do more than simply install themselves to run at boot-they alter the computer's operating system. (Operating systems are large collections of programs, text files, shared libraries and other information that do not have a strict technical definition, but it is well within the capabilities of companies such as Microsoft and Apple to provide a list as to the names of the files that are in each release their operating systems.) Seeing this icon would give users a reason to ask questions. More likely, forcing this kind of disclosure would simply end the practice on the part of developers.

Monitor: Keeps Track of What The User is Doing
Most programs mind their own business. But some software watches the user's keystrokes, spies on the screen, or monitors the Web pages as they are downloaded—even when another program is running in the foreground of the user interface and has the keyboard focus. Other kinds of monitoring include watching activity in the file system, making copies of documents as they are printed, or simply noting when the computer is idle and when it's in use. (Screensavers are prime examples of such programs, and the APIs created for screensavers have been used in the past to write keyboard sniffers and other kinds of Trojan programs.) The key here is that personal information is being captured by a program when the user thinks that the program is not listening. Many AOL Instant Message users, for example, appear to be surprised when they find out that whether or not their computer is "idle" is transmitted to all of their AIM "buddies." Google Desktop Search monitors the computer's file system and scans through all of the user's files. Clicking on the icon would reveal how the monitoring took place, the purpose of the monitoring, and relate to what uses the information would be put. Finally, instructions would be provided for turning off the monitoring.

Pop: Displays Pop-Ups When Running In Background
Many computer users are annoyed by pop-up windows containing advertisements. Users are annoyed by both the content of the windows and by the fact that they interrupt other activities. On the other hand, programs like Microsoft Word display pop-up dialogues in response to commands like "File Open." What distinguishes these two classes of pop-up windows is whether or not the program displays the pop-up when it is active, or when it is running in the background. To be sure, not all pop-ups are bad. Some calendar programs will display pop-up windows when an alarm goes off, even if they are not the active program. By rights, those programs would also need to carry this icon.

Remote Control: Lets Remote Users Take Over The Computer
In theory, any program that's running on a computer can take it over and execute commands on behalf of others. In practice, only very few programs explicitly incorporate remote control functionality. Programs that have this capability should be labeled.

Self-Updates: This Program May Change Its Behavior Unexpectedly
One of the most important techniques for software vendors to deal with persistent computer security problems is to have their programs automatically update themselves with code downloaded from the Internet. But the ability to self-update can also be a boon to makers of spyware: it allows them to add new, nefarious capabilities to programs that have previously been examined and found to be benign. This is an example of an icon that is neither *good* nor *bad*, per se, but that merely documents behavior that can otherwise be very difficult to detect. The icon tells users that the behavior of the program can radically change without any input from the user.

Stuck: Cannot be Uninstalled
Some programs are truly impossible to dislodge. These programs are typically operating system updates, but it is easy for a clever programmer to make uninstallable spyware as well. Consumers should be informed that there are some programs for which there is no going back.

With the icons would need to come rules for their use. For example, some of today's click-through license agreements say that the user implicitly agrees to any changes in the license agreement unless those changes are "substantive." But what is substantive? Once a labeling regime was in place, a substantive change could be legally defined as a change that results in a change of icons. An example of such a substantiative change would be for a self-updating program to download and install a remote-control feature. The law could then require that this sort of change would require new consent on the part of the user. Figure 8-7 shows how the icons might be added to the Windows Add/Remove panel, while Figure 8-8 shows how they could be added to the Google Toolbar license.

8.4 RFID on Consumer Items: The "RFID Bill of Rights"

The Electronic Product Code (EPC) is a system that applies Radio Frequency Identification (RFID) technology to the task of supply chain tracking and supermarket check-out. Proponents of RFID describe a world where small EPC tags will be built into the packaging of consumer goods much in the way that barcodes are placed on packages today. These tags, combined with readers strategically placed throughout the supply chain and high-availability networked database, would permit the tracking of consumer goods from the point of their manufacturer to their disposal. EPC could allow manufacturers to automatically detect pilferage at the factory, diversion from one market to another, and supermarkets that are running low on inventory. If the EPC is embedded in the consumer good itself, rather than the packaging, EPC could allow for the easy identification of items by the blind and the automatic segregation of goods containing hazardous materials at trash disposal facilities.

Unlike optical barcodes, EPC codes can be read at a distance and through materials. More over, whereas UPC barcodes only identify the type of consumer item purchased, EPC codes can be used to identify the actual item—revealing that one razor was bought at a Safeway at 11:15am on July 12, 2005, and another was bought at a Store24 at 7:15pm on July 13. By combining checkout information with credit-card payment records or biometric identification systems (for example,

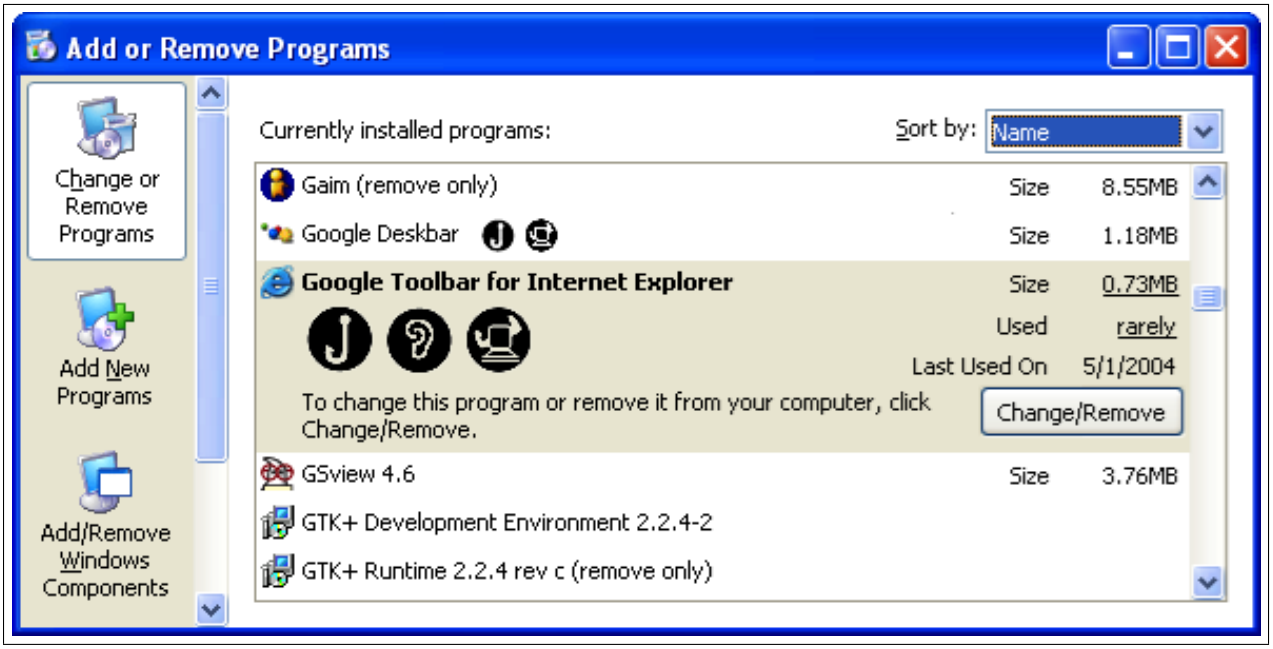

Figure 8-7: The Windows Add/Remove panel, modified to show software icons. The Gaim, GSview, and GTK+ programs do not have icons because they do not engage in any icon-worthy behavior. *Simulated screen shot.*

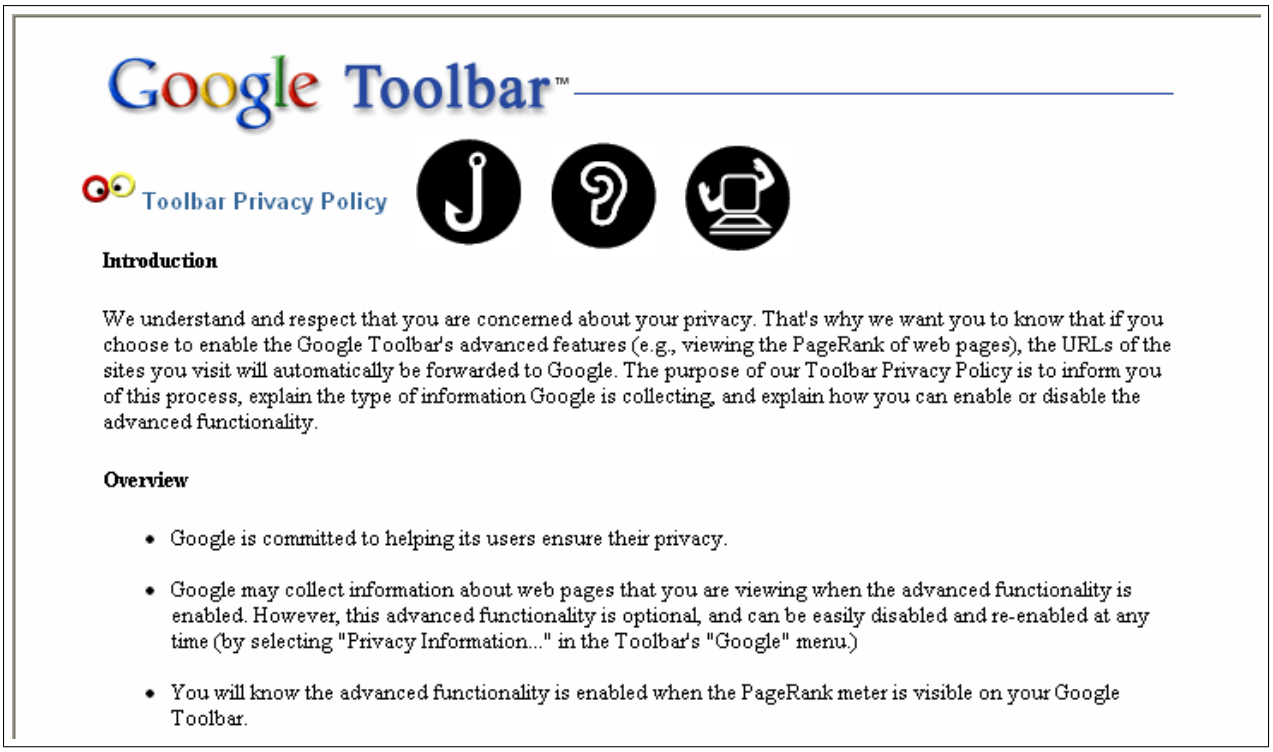

Figure 8-8: The Google Toolbar license, annotated with software icons proposed in this chapter. *Simulated screen shot.*

"Users of RFID systems and purchasers of products containing RFID tags have:

1. The right to know if a product contains an RFID tag.
2. The right to have embedded RFID tags removed, deactivated, or destroyed when a product is purchased.
3. The right to first class RFID alternatives: consumers should not lose other rights (e.g. the right to return a product or to travel on a particular road) if they decide to opt-out of RFID or exercise an RFID tag's "kill" feature.
4. The right to know what information is stored inside their RFID tags. If this information is incorrect, there must be a means to correct or amend it.
5. The right to know when, where and why an RFID tag is being read."

Figure 8-9: The RFID Bill of Rights.[Gar02]

video surveillance data), EPC codes can be matched up with individual human beings.

RFID privacy issues are difficult to solve with cryptography because on-tag computation is extremely limited as the result of low silicon and power budgets. Usability poses other constraints: although Weis [Wei03] proposes a "hash lock" for preventing the unauthorized reading of tags, it is unclear how untrained consumers would be able to effectively manage such locks. Jules, Rivest and Szydlo propose the use of an RFID "blocker tag" that would give consumers who carry it a zone of privacy to prevent the reading of other RFID tags.[JRS03] However, many find it unpalatable to suggest that consumers will only be free of covert RFID monitoring if they choose to purchase and carry such a device.[Pri03]

Two different strategies for linking RFID chips to databases further complicates the privacy analysis. Although simple RFID chips contain only a serial number, so-called second-generation RFID chips can contain rewritable persistent memory. Thus, a chip may contain a substantial amount of personally identifiable information—information that may not be as easy to audit or correct as information stored in a conventional database. Indeed, this amount of rewritable storage could even constitute the kind of "secret database" that the Fair Information Practice is designed to prohibit.

One approach to addressing RFID privacy issues is to increase the visibility of tags, readers, and their purpose through the application of the Code of Fair Information Practice to RFID technology. This approach is called the "RFID Bill of Rights" and consists of five principles, shown in Figure 8-9.

These rules could be the basis for future regulatory measures to address the use of UPC-enabled products in the future. Regulation is a good complement to technical measures: since neither regulation nor technical measures can be 100% effective, the combination of the two has a greater chance of protecting privacy than either one by itself.

RFID Legislation

Although there has been some interest in legislation that would curb the use of RFID in consumer products, most of this legislation appears to have stalled.

For example, in February 2004 California state Senator Debra Bowen introduced Senate Bill 1834 that would require businesses and agencies to notify people that an RFID system was in use,

and require that retailers detach or destroy RFID tags on merchandise before it leaves the store's premises.[Gil04]

However, the California RFID privacy bill failed to pass committee when it was heard in June 2004. Opponents of the bill, including Hewlett Packard, the American Electronic Association, the California Chamber of Commerce, the California Grocers Association, the California Retailers Association, and Grocery Manufacturers of America, argued that the legislation was premature "and that the bill should not precede the actual installation of RFID in businesses and libraries." [Swe04]

Other US states are exploring RFID. For example, the Virginia Joint Commission on Technology & Science included an examination of RFID issues in the Commission's 2004-2005 "Work Plan." [Vir04]

Roberti has argued that legislation isn't the answer to the RFID privacy problem until a privacy problem has been shown to exist through poor corporate practices.

> "One of the best things governments could do is to help educate consumers about what RFID is, what it can and can't do and what information could be collected. If consumers understand the technology, they will let retailers know what they are—and are not—willing to accept by deciding where to shop. Some might say that retailers will use the technology secretly to spy on their customers. It's possible, but when that company is exposed—and it will be exposed—it will damage its credibility and lose customers, and that will be a lesson to other retailers."[Rob04a]

Roberti asserts that consumer advocates really fear "that people will passively accept whatever retailers choose to do. What they are saying is they know more about what's good for the consumer than the consumer does. I give people more credit than that."

Unfortunately, the evidence from the food labeling studies cited in Section 2.6.2 indicates that experts *do in fact know more about what's good for consumers than consumers do.* Rather than allow the tools for covert surveillance to be created and deployed, it makes more sense to pause and examine what problems could be averted wit a simple disclosure regime.

8.5 Conclusion

This chapter reviews two policy proposals and shows how both are based on two general patterns of US technology regulation. The first of these patterns is the use of standardized terminology. The second is that significant deviations between what the technology actually does, and likely mental models of what the technology does, must be disclosed to users and potential users of the technology.