CHAPTER 1

Introduction

It is widely believed that security and usability are two antagonistic goals in system design. This thesis argues that there are many instances in which security and usability can be synenergistically improved by revising the way that specific functionality is implemented in many of today's operating systems and applications.

To be sure, today's systems often force users into a dilemma of choosing between security and usability. But this is frequently a fool's choice, for a system that is secure but not usable will not be used, while a system that is usable but not secure will eventually be hacked and rendered unusable.

My thesis is that there is no inherent conflict between creating systems that are secure and systems that are usable. Instead, I argue the reverse: **Usability and security can be made synergistic by redesigning systems with specific principles and through the adoption of well-defined patterns.**

In some cases it is possible to simultaneously increase usability and security by revisiting design decisions that were made years ago. In other cases it is possible to align security and usability by changing the regulatory environment in which the computers operate. To these ends, this thesis presents principles and patterns that cover three specific areas: patterns that prevent the accidental release of confidential information through remnant data; patterns that promote secure electronic messaging; and patterns that reduce the danger of covert monitoring through software and radio frequency identification (RFID) systems.

Throughout this entire body of work, the goal is not to make security *invisible*, but to make security a natural result of normal computer operations. The goal is not to make systems that are *theoretically securable*—the goal is to make systems that are *actually secure* when they are used in common scenarios.[Tog05] To accomplish this goal, techniques will be presented that make security *automatic*, *understandable*, and *auditable* by non-expert computer users.

1.1 Security vs. Usability: The Need for Design Patterns

The need to make it easier for end-users to securely operate their own computers is increasingly seen as one of the leading security problems of our time. For example, at the 2003 Computing Research Association's conference "Grand Challenges in Information Security & Assurance"[CRA03], the need to create better end-user security controls was identified as one of four "grand challenges" facing computer security researchers. In 2005, the President's Information Technology Advisory Committee identified improved techniques for end-user security as one of nation's foremost priorities for cybersecurity research.[Pre05]

The basic argument of this thesis is that common errors in system design, computer user interfaces, and interaction design can lead to common errors in secure operation. By identifying and correcting these errors, users can naturally and automatically experience more secure operation.

In principle is this not a new approach. Much security research over the past twenty years tried to increase system security by identifying common flaws and errors, and then proposing solutions that could be used in a variety of different circumstances. But while this cookbook-like approach has been applied to common security problems such as buffer overflows[CPM$^+$98] and the transmission of passwords by cleartext protocols[FK96, Ylo96], it has not previously been applied to techniques for promoting secure Human Computer Interaction (HCI).

1.1.1 Principles and patterns

Saltzer and Schroeder introduced the concept of using *design principles* to improve the security of computer systems in the classic article, "The Protection of Information in Computer Systems."[SS75] Those principles crystallized years of experience resulting from the design and implementation of the CTSS and Multics operating systems. They provide designers with a tool for applying the lessons of those operating systems to future projects. They also provide a conceptual framework for teaching the lessons to the next generation of designers and programmers—which is fundamentally the only way to ensure that the knowledge is not lost.

But for all of their power, there is no obvious way to translate design principles into concrete code or even into a specific element of system design. Because of their generality, design principles are necessarily subject to interpretation.

Design patterns overcome this problem by providing specific solutions to commonly occurring problems. Good patterns are like recipes: they tell you what elements they require and then provide step-by-step instructions on how to use them. The best patterns also include context information about when they are applicable, when they are not, what they accomplish, and how to adapt them to a specific situation. To continue the food analogy, these patterns are like the detailed plans for a complex dinner party.

A brief history of patterns

Patterns are used in many human endeavors that require a combination of skill and training. For example, Schmidt *et al.* note that textile designers choose fabric by its pattern and create patterns for the design of clothes; pilots fly in patterns; and engineers make use of common circuit patterns in the design of electronic devices.[SFJ96]

Architect Christopher Alexander pioneered the recognition, naming, and use of patterns while working on urban planning in the 1970s. [Ale79, AIS77] In the late 1980s and early 1990s a number of computer scientists working in the field of object-oriented design discovered Alexander's work. They saw a strong similarity between Alexander's reusable architectural patterns and class libraries made possible by object-oriented languages such as Simula, SmallTalk, and C++: just as Alexander's patterns were generic architectural solutions, the class libraries could be thought of as generic programmatic solutions.

Although many of the patterns presented in the 1990s could more properly be thought of as well-justified class libraries, patterns can be developed at considerably higher levels than mere code. Schmidt *et al.* argue that the real value of patterns is to encapsulate knowledge and understanding, making it easier to teach and deploy solutions.[SFJ96] Patterns make it possible to reuse successful practices, to reason about what is done and why, and to document abstractions other than algorithms and data structures. For example, the patterns-based approach has been applied to a wide variety of problems, including Avionics Control Systems [Lea94], System Reengineering [SP98], and even Risk Management.[Coc05]

"Best practices" and patterns for computer security

At its very heart, computer security is an engineering discipline. It is impossible to have a computer system that is completely secure. Instead, security practitioners and the organizations that employ them must analyze their risks, the costs of proposed security measures, and the anticipated benefits.

While practitioners have long understood the theory behind conducting a formal cost/benefit analysis, it is exceedingly difficult to apply such approaches to security because the risks frequently defy measurement or estimation. What is the risk that a piece of code will contain a buffer overflow? What is the risk that an attacker will discover the flaw and be in a position to exploit it? What is the cost of a successful exploit? Unable to come up with hard numbers, in the 1990s an alternative approach called "best practices" emerged. Briefly, this approach seeks to create a standardized catalog of the security practices that are used in an industry, and then employ that catalog as a kind of security checklist. Although it has been broadly applied to computer security[SAN04], the best practices approach has been applied to many other security-related fields, from financial crimes[FC99] to terrorism.[Jen97]

The problem with the best practices approach—aside from the fact that the practices suggested are usually "minimal" rather than "best"—is that mere catalogs do not explain how practices fit together or what each practice accomplishes. Understanding what makes these practices the "best practices" is literally left as an exercise for the reader.

Security patterns are an attractive supplement or alternative to best practices. Like best practices, patterns provide a cookbook-like approach that can be used by those less-skilled in the field. But unlike best practices, patterns can also provide a framework (a pattern language) that can aid in teaching, understanding, and even in formal system analysis.

Smetters and Grinter specifically suggested that the field of usability and security "would benefit from creating and using security-related idioms or 'patterns' similar to the software use patterns common in other areas of development. These could help developers less sophisticated in the use of security technology to understand how to incorporate it more effectively into their applications."[SG02] Until now, the research community has largely ignored this suggestion.

1.1.2 Why patterns are needed for solving the usability and security problem

Patterns are an especially useful tool for solving multidisciplinary problems such as the alignment of security and usability.

Both security practitioners and usability specialists have long argued that what they bring to the development process—be it security or usability—cannot be easily added to a completed system as an afterthought. Instead, security and usability must be designed into systems from the beginning.

Here, then, is the origin of the usability/security conundrum: very few developers are trained in *either* security or usability, let alone both. Very few product teams have a security specialist or a usability specialist, let alone one of each. There is a universe of developers with all kinds of skills—graphics, microcoding, device drivers, compiler design, and so on. Given such a universe, and given that security and usability are different skill sets, the number of individuals or teams that have in both security and usability is likely to be quite small. (Figure 1-1)

By providing pre-packaged solutions to common design problems, patterns can address this deficit. What's more, patterns can boost innovation by making it possible for designers to build upon the work of others in the field of usability and security (HCI-SEC[1]). Many patterns have been developed and deployed over the past 20 years that have dramatically increased the usability of modern computers; examples of these patterns include copy-and-paste, drag-and-drop, and even very specific patterns such as the highlighting of misspelled words. Likewise, security patterns such as the use of the Secure Socket Layer (SSL) to "wrap" cleartext protocols and Email-Based Identification and Authentication for resetting passwords have allowed developers untrained in security to nevertheless increase the security of their systems. By creating and publicizing patterns that align security and usability, it is reasonable to expect that progress will be made in this area as well.

1.1.3 Principles and patterns for aligning security and usability

This thesis shows that patterns which simultaneously promote security and usability can be developed in a variety of areas. For each set of patterns studied, it shows that these patterns can be applied to multiple cases—bolstering the claim that these are general design patterns, rather than a specific technique that works in but a single situation.

The patterns that are presented are grouped into three specific areas:

- **Patterns for User Visibility and Sanitization**

  These patterns are aimed at eliminating various kinds of "hidden information" that is left-behind on computer hard drives, in applications such as web browsers, and in complex document files. Taken together, the patterns overcome a common problem in today's computer systems: that the commands for performing "delete" and "erase" operations frequently remove

[1]HCI-SEC is a commonly used shorthand to describe the research field concerned with the alignment of security and usability. The term HCI-SEC is a combination of the acronym HCI (Human Computer Interaction) with the abbreviation SEC (Security). Whitten popularized this term when she created the HCISEC group on Yahoo! Groups.[Whi00]

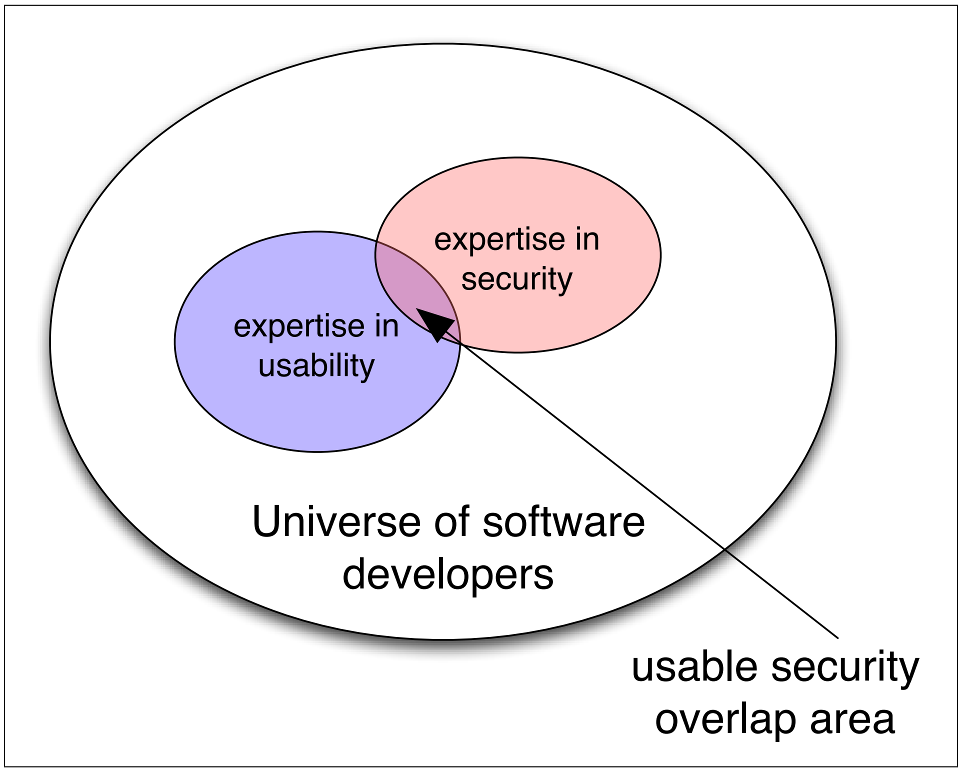

Figure 1-1: The usable security overlap area

visible indication of the information's presence, without actually removing the information itself.

Chapters 3 and 4 discuss this issue at length, showing that left-behind data is pervasive on today's computer systems, and that the failure of modern systems to properly sanitize media and files has led to many cases in which confidential information was compromised.

Until now, the most common solution to the problem of data left behind was education: users were warned of the risk and then told of specific third-party programs that could wipe or otherwise sanitize computers, web browsers, and document files. For example, in 2004 the Federal Trade Commission and other regulatory bodies adopted rules requiring that businesses acquire technology and train their employees in the use of such programs to ensure that computers containing "consumer information" were properly sanitized prior to being disposed.[Com04b]

A complementary approach, proposed in this thesis, is to rework the operating system file deletion system calls so that the blocks corresponding to deleted files are actually overwritten. But as this thesis will show, simply changing the behavior of these system calls is insufficient for simultaneously promoting end-user security and usability. Unless all of the identified patterns are implemented, specific cases will remain through which end-user security or usability will be readily compromised.

- **Patterns for Secure Messaging**

Public key cryptography was invented by Diffie and Hellman in 1976 for the explicit purpose of letting people exchange email without first requiring the exchange of shared secret keys.[DH76] Yet the experience of the past 29 years is that public key cryptography has often just replaced the old problem of key distribution with a new problem—the problem of key certification. Kohnfelder's thesis[Koh78] showed that digital certificates could be used to distribute keys signed by a trusted third party—what is today called a Certification Authority. But while CAs have worked within organizations and for the certification of some business web sites, the CA approach has generally failed to certify the keys of end-users.

This thesis investigates two aspects of the secure messaging problem: do-not-reply email sent from large organizations, usually in response to some kind of e-commerce event; and person-to-person email, such as email exchanged between co-workers.

In the case of do-not-reply email, this thesis argues that considerable improvements in overall security could be realized today if such mail were signed with S/MIME digital signatures. This argument is supported through a detailed analysis of mail clients, web mail systems, and through the results of a survey that was conducted of 469 Amazon.com merchants. Finally, the techniques that are specifically needed for mail signatures are clarified in a series of patterns.

To support person-to-person secure messaging, this thesis presents patterns that refine an alternative approach for securing public keys called *Key Continuity Management*[Gut04b] (KCM). This approach replaces the Certificate Authority with a client-side database that is used to maintain a binding between the key and the identity for which it was used. This binding is then verified on each subsequent use of the key, and the user is alerted if the binding changes. KCM is precisely the security model introduced by Ylonen in the SSH application[Ylo96], although the term itself was coined by Gutmann.[Gut04b]

This thesis refines and analyzes Key Continuity Management, showing that it offers security guarantees that are similar to and in some cases identical to the security guarantees provided by traditional CA-based systems. It then presents the results of a user test of 43 subjects, showing that Key Continuity Management can defend against a variety of spoofing attacks that are affecting email users today.

- **Patterns for Promoting Overall Secure Operation**

  The final patterns presented in this thesis are a collection of specific techniques and practices which are designed to further promote overall secure operations without cost to usability. These patterns are supported by a comprehensive review of the HCI-SEC literature and a consideration of non-technical factors that have been shown to impede the usability of security technology.

In addition to these patterns, this thesis also presents five principles upon which these patterns are loosely based.

1.2 Computer Security at the Crossroads

Why has usability only recently emerged as an issue of concern for the security community?

One possible explanation for this recent interest is the changing nature of the world's computing environment. Poor security measures have historically been self-correcting events. Militaries and corporations that skimped on security suffered accordingly. Computer systems, meanwhile, were administered by a relatively small group of technically proficient individuals. In such an environment, systematic problems in security usability could be overcome through training, the use of consultants, or even the threat of punishment.

The proliferation of high-performance computers with high-speed Internet connections has changed the calculus of security. Most of these systems are used on a regular basis, but their security is not actively monitored. Once compromised, they become launching points for spam, denial-of-service attacks, and even illicit web hosting. The shift is significant: Whereas poor security measures were once the most damaging to the owner of the poorly administered system, they are now more damaging to others on the Internet or to society as a whole. Indeed, the owner may not even directly suffer from using an infected host unless the owner's ISP notices the infection and terminates the computer's service.

To put this in biological terms, it was once the case that security threats were spot attacks that put evolutionary pressure on computer systems and the organizations running those systems to be secure or to die—it was a case of "evolution in action."[NP81] Today's security threats are more closely modeled as communicable diseases that weaken but do not kill their hosts—at least, not until the infectious agents have reproduced and spread to other hosts.

In September and October 2004, technical experts working for America Online examined 329 home computers and found that 20% of them were currently infected by a virus; in interviews, 63% of the computer owners acknowledged that their computer had been infected in the past. A whopping 80% of the systems were also infected with adware or spyware (a topic that will be addressed in 8.3). And even though 85% of the machines surveyed had some kind of antivirus systems installed, 67% of those systems lacked up-to-date antivirus signatures and were thus ineffective against the latest threats. Ironically, 70% of those who participated in the survey believed that they were safe from viruses and online threats—many people in that 70% were mistaken. [Rob04b]

All of the viruses identified in the AOL study were, by definition, viruses that were recognized by existing antivirus software: the fact that the infections were not identified indicates that the current model of defending against hostile software simply does not protect many home users. This observation is echoed by Gutmann, who concludes that the Internet's current plague of viruses, worms and Trojan horses is not because of novel buffer overflows, rootkits, and hacks against Microsoft's operating systems, but because "existing mechanisms are too hard to use" and "existing mechanisms solve the wrong problems."[Gut04b]

1.2.1 Computer security: a definition

Phrases like *security breaches* and *computer security* mean different things to different people. These days one might be tempted to define a secure computer as a computer that is not susceptible to attack over the network. By this definition, a laptop that is not plugged in to a network could be thought to be "secure." But an owner who leaves such a laptop unattended at a hotel bar may be disappointed to find that his or her "secure" computer has been stolen. Clearly, being able to withstand an attack over a network is not the only measure of security.

One definition of security that seems to be widely accepted is this:

> **Computer Security:** "A computer is secure if you can depend on it and its software to behave as you expect it to."[GS91]

This definition may seem overly broad, but it does force the practitioner to focus on such practical goals as continuity of operation in addition to classical goals such as prevention against unauthorized disclosure.

1.2.2 User models for the 21st century

There appears to be no historic model for users of computer systems other than the tautology that "computer users are people who use computers."

Writing in 1979, Morris and Thompson discuss the predilections of PDP-11 Unix users to pick passwords that are easily guessed, but they didn't say anything about who those users actually were.[MT79, p.596] In 1987 Brian Reid stated that "programmer convenience is the antithesis of security, because it is going to become intruder convenience if the programmer's account is compromised."[Rei87, p.105]. However, it is clear from a reading of Reid's essay in *Communications of the ACM* that when he wrote the word "programmer," Reid actually was referring to people who were system managers of Unix systems—people who in 1987 were frequently programmers like Reid.

In fact, computers have had a wide range of user populations for decades—users consisting not only of computer scientists and researchers, but also secretarial staff, emeritus professors, and even the school-aged children of researchers who had access to computer systems through home terminals.

Nevertheless, there was almost certainly an implicit user model at work. That model saw the computer user as an able-bodied, moderately educated English speaker, able to read, write, see, hear, and type. This user model was so entrenched that only within the past decade have efforts at accessibility and internationalization produced computers that can be readily used by people with disabilities or who do not speak some amount of English.

In recent years we have seen the emergence of an expanded user model that includes users with both physical and mental disabilities, users who do not speak English, users who are not literate, and—in many cases—users who are not even human. It is hypothesized, for example, that within a few years much of the information accessed over the World Wide Web will be accessed by agents, robots, and other kinds of non-human savants.

This expanded user model has significant implications for computer security, not the least of which is that usability will be far more important in the future than it has been in the past. Many observers have noted that one of the things that makes security systems hard to use is that there are many special cases which can only be properly handled with skill and training: if the cases were not special, then their handling could be automated and the security problem would go away.

But whereas others have argued that the solution to this expanded user model is increased efforts directed towards user education—for example, user interfaces that teach security concepts—I be-

	Dollar Loss	
Attack	2004	2003
Computer Virus	$55,053,900	$27,382,340
Denial of Service	$26,064,050	$65,643,300
Theft of Proprietary Information	$11,460,000	$70,195,900
Insider Abuse of Net Access	$10,601,055	$11,767,200
Abuse of Wireless Network	$10,159,250	
Financial Fraud	$7,670,500	$10,186,400
Laptop Theft	$6,734,500	$6,830,500
Unauthorized Access by Insiders	$4,278,205	$406,300
Telecom Fraud	$3,997,500	$701,500
Misuse of Public Web Application	$2,747,000	
Web Site Defacement	$958,100	
System Penetration	$901,500	$2,754,400
Sabotage	$871,000	$5,148,500
Active Wiretap		$705,000
Passive Eavesdropping		$76,000

Figure 1-2: Security threats facing US organizations and reported dollar losses, summary data from the 2003 and 2004 CSI/FBI Computer Crime and Security Surveys. Blanks indicate that the category was not included in the annual survey or that no loss was reported. [CSI03, CSI04]

lieve that the correct solution to the user diversity problem is to redesign our systems so that secure operation emerges organically when users pursue their existing goals.

1.2.3 Threat models for the 21st century

It is somewhat easier to quantify the historical threat model, if only because there is a rich literature of computer attacks from which to draw.

As will be discussed in Section 2.3.1, Clark and Wilson have argued that computer security researchers have historically considered threats of data theft to be of primary concern and technologies for preventing disclosure of confidential material to be of preeminent importance.[CW87] This concern is traced to the US intelligence community, which was traditionally one of the principle funders of computer security research.

But while the user model is expanding, the threat model is changing. With each year, it seems, new vulnerabilities are discovered and exploited, forcing a continual reassessment of security strategies and expenditures. Few of these new threats represent the kind of data disclosure that was of concern to funding agencies. For example, the CSI/FBI 2004 Computer Crime and Security Survey defined 13 types of attacks or computer misuse resulting in direct financial loss to the survey's respondents; three of these categories—abuse of wireless networks, misuse of public web applications, and web site defacement—didn't even appear in the 2003 survey. On the other hand, two areas of demonstrated monetary loss in the 2003 survey—active wiretap and passive eavesdropping—did not appear in the 2004 survey.[CSI03, CSI04] Figure 1-2 summarizes these categories and the reported dollar loss for the years 2003 and 2004.

The majority of the threats identified by the CSI/FBI survey involve a failure of security according to the definition presented in [GS91]. A computer that is "secure" is not a computer that allows the theft of proprietary information, yes, but it is also not a computer that is susceptible to denial of service attacks or computer viruses, it is not a computer that can be abused by insiders, and so on. This is an expanded threat model that clearly considers attacks other than disclosure to be serious attacks that are worth preventing.

1.3 Why Have Security Specialists Failed to Address Usability?

With the exception of Saltzer and Schroeder's classic 1975 article[SS75] and a handful of others, [Bor96, MT79, GLNS93, Rei87, Kar89, ZS96] an in-depth examination of the computer security literature shows that the security community largely ignored usability issues until the late 1990s. Likewise, an examination of the usability literature shows that the usability research community did not actively research usable security solutions during that same period.

One explanation for the failure of security specialists to address usability issues is that security and usability have traditionally been been as being mutually antagonistic. If true, then there would be little conceivable motivation for one community to work on issues that would appear to have a contradictory goal.

After some reflection, this explanation is clearly wrong. The research community is often interested in technical trade-offs—for example, space-time trade-offs in algorithm design, or the difficulty of performing high-strength cryptography on relatively slow and under-powered microprocessors. The perceived antagonism between security and usability could have been taken as a challenge and *stimulated* research, rather than deadened it.

A more plausible explanation is that researchers were busy exploring a wide range of questions in both specialties that could be addressed without the need to become familiar with another discipline. Development of secure operating systems and new encryption technologies was so demanding that it left little time to work on usability issues.

Yet another explanation is the possibility of a culture clash or personal animosity between individuals who engaged in security research and those who engaged in usability work.

1.3.1 The emphasis on cryptography

It is possible that the heavy emphasis on cryptographic techniques for protecting information in computer systems is one of the elements responsible for lack of general attention to the issue of usability.

Cryptography is a problem that is both difficult and deep: it is easy to imagine that the emphasis on these important problems have diverted time, attention, and financial resources from other areas of computer security research. For example, Morris and Thompson noted that there is a tendency for computer scientists to focus on intellectual problems that are mathematically interesting to the exclusion of messy real-world problems which must be solved in order to actually increase operational security.[MT79, p.594]

There are other aspects of cryptography in particular which poses significant usability challenges. As discussed in Section 8.2, cryptography's terminology is complex and frequently used inconsistently. Keys have a unique usability problems: a single flipped bit can be the difference between data that can be recovered and data that is lost forever. And revealing a key can have profound impact, gives keys security properties that are very different from the house keys after which they are named, and more similar to the security requirements required for a bio warfare pathogens.

1.3.2 Industry's emphasis on bug fixing, rather than secure design

A second factor that may be responsible for the decreased stature of usability research in the field of computer security is the industry's emphasis on bug fixing rather than secure design. The standard approach for running secure systems is to make sure that virus definition files are up-to-date and that all of an operating system's latest patches and bug-fixes are downloaded and installed on a regular basis. As a result, much research to date has been on techniques for automatically implementing or eliminating these tasks, rather than looking for other opportunities to change the underlying operating system to promote more secure operations overall.[SAN04]

Bug fixes and antivirus systems are a tactical, short-term approach to strategic, long-term problems. They are band-aids rather than attempts to address underlying diseases. Training people to cope with software that is difficult-to-use, rather than redesigning that software, is another such tactical response. Nevertheless, it is a profitable opportunity that detracts attention from the fixing of underlying causes.

1.3.3 Emphasis on new tools, rather than secure operations

Kim and Spafford have argued that many organizations are overly focused on security tools when greater benefits can be achieved by focusing limited resources on controls and process improvement. They advocate a methodology known as "Visible Ops" in which Information Technology operations are broken into discrete steps. Each step is a project, "with a clearly defined objective and exit criteria."[KBS04] Steps are ordered, each building on the previous step. Steps are catalytic, each resulting in a benefit to the organization. They are sustaining, in that they create enough value for the organization so that there is reason to keep each step even if a subsequent step is abandoned. Finally, Visible Ops steps are auditable.

Kim, the co-author of the Tripwire integrity management tool,[KS94] has had an uphill battle in his efforts to persuade organizations to focus their attention on secure operations. It's easy for an organization's management to allocate budget to purchase new tools, with the hope that those tools will improve overall security. It is much more difficult for an organization to commit to changing its internal practices and procedures to an approach that will almost surely increase short-term costs—even if long-term costs are significantly reduced.

1.3.4 Perverse market incentives

Complicating the problem of developing computers that align security and usability are a number of perverse market incentives that paradoxically favor the development and deployment of systems that are unwieldy and difficult to manage.

Innovation in the security marketplace is frequently driven forward by small companies that develop so-called "point solutions" that address a specific problem. Over time other companies develop their own versions of these point solutions, the product category matures, and eventually the technology is built into the operating system (where the phrase "the operating system" refers both to the actual operating system and to the suite of programs that accompany the operating system.) This is the path that has been followed by firewalls, junk mail filters, spyware scanners, and now by antivirus systems.

There are many reasons why this model of point solutions does not produce security systems that are easy-to-use:

- Third-party point solutions typically need to be separately purchased and installed.
- Even when third-party solutions are pre-installed by hardware manufactures or IT departments, there is still a marketing incentive to make sure that the these products are noisy. The solution must announce its presence and give the user the impression that it is active and protecting the user's interests. If the solution is silent, no one will know that the solution is present.
- Third-party security solutions are necessarily created by different development teams and shipped separately from the systems that they intend to secure. As a result, they frequently have different user interfaces and may have problems integrating with some operating system configurations.
- As both the number of third-party solution providers and the number of solution categories increases, the number of possible combinations and permutations increases geometrically. Individual users may deploy solutions that were not tested and which may not produce good results, causing the tools themselves to contribute to system failures and a lack of usability. For example, a user may install and run antivirus systems from Symantec and F-Secure, combine this with pop-up blockers from Google and Microsoft, and throw in two home-firewalls for good measure. The result may well be a computer system that is not usable. Such combinations can happen even in a corporate environment, where deployment configurations are not consistent or end-users seek to supplement the security systems issued by their IT support staff.

David Clark tells an amusing story in which a friend discovered that her email was no longer filtered for viruses after she enabled the "POP over SSL" feature in her mail client. Her antivirus system had been screening her email through the use of a POP proxy.[Cla03] Once SSL was enabled, the POP proxy couldn't examine the contents of the mail stream because it was cryptographically protected against such man-in-the-middle attacks!

1.3.5 Difficulty of conducting usability research

When computer security specialists are motivated to conduct research in the field of HCI-SEC, a problem that they immediately face is that HCI-SEC research frequently requires the experimenter to experiment on human beings in the form of user studies. The skills required for conducting effective user studies, while learnable, are unlike other skills required for success in computer science. In a university or government environment, user studies are further complicated by the need to comply with federal regulations that govern the use of human subjects. The added paperwork can introduce delays and scare off casual research that might lead to insightful new discoveries.

Researchers who are seriously interested in conducting user studies in a federally approved manner have gone on to discover that it is difficult to test the usability of a security tool under realistic conditions. Although one approach is to subject users to hypothetical attacks, the attacks need to be carefully calibrated so that they are not too weak and not too strong.[SG02] Many usability issues only emerge when systems are used infrequently, an approach hard to replicate in a lab. Sasse notes that in many cases it is necessary to followup a user test with a second test performed after the initial training—otherwise "you have no data."[Sas04a] But such protocols add to study expense and difficulty.

1.3.6 The authentication conundrum

For security researchers who are interested in usability concerns and who have the ability to conduct user tests, another factor is the strange attraction of HCI-SEC researchers to the authentication problem—invariably resulting in the exclusion of other HCI-SEC issues because of limited research time, attention, and budgets.

Authentication in computer systems is commonly described as being based on "something that you know" (*e.g.,* a password), "something that you have" (a token or smart card), or "something that you are" (a biometric). Authentication systems frequently fail because they are actually based on something that you have forgotten, something that you have lost, or something that you no longer are. Performance-based biometrics (*e.g.* keystroke dynamics) fail when they are based on something that you could once do well but can no longer do, or something that other people can do consistently, but you do erratically.

Research on authentication is tremendously important. Without authentication, a computer system frequently has no basis for determining if access should be granted or not. Even capability-based systems that provide access without authentication need to have some kind of system for deciding who gets the capabilities. What's more, practically every modern computer user needs to authenticate and re-authenticate themselves multiple times throughout the day. As the current state of authentication systems is generally deplored and ridiculed, any advances in this field should have a huge social benefit.

But authentication is a particularly difficult area of research because today's authentication systems do not fail gracefully. If a user can only remember eight characters of a nine-character password the computer does not allow the user limited access to the system's less critical files: access is simply denied. This is very different than authentication in the offline world. A bank, for instance, might allow a person who has only weakly authenticated herself to withdraw a few hundred dollars but might require substantial proof of both identity and authorization to withdraw several hundred thousand dollars in cash.

Not surprisingly, research into authentication has taken up an astounding amount of resources over the past three decades but has produced few tangible results. For example, despite the tremendous amount of effort that has been spent developing certificate-based authentication systems for SSL, the majority of web sites operating on the public Internet appear to base their authentication on usernames and passwords, and not on client-side certificates. This is the same authentication technology that was used by CTSS in the 1960s! Even worse, organizations like VeriSign, Thawte, and even MIT that actually issue client-side SSL certificates will frequently issue them to anyone who has a correct username and password. (In the case of MIT, the "password" consists of the student's Kerberos password and their MIT ID number.)

The depth of the authentication conundrum is evidenced by disagreement on such fundamental questions as password expiration policy, whether or not passwords are even a good idea, and whether systems that supplement passwords with tokens can or should be deployed to a large user base.[Ric05] Password use rules are inherently self-contradictory: many policies imply that users must pick passwords that are impossible to remember (because they contain no patterns and nothing of personal significance to the user) and then avoid at all costs the security risks inherent in writing these passwords down!

Complicating matters is interest in biometric technology, which is simultaneously seen as a promising technology for authentication, a technology with inherent usability problems that has never been deployed on a large scale, and a dangerous technology for social control if placed in the hands of the government.[Cov05]

While this thesis notes the depth of the authentication problem, it will attempt to avoid addressing the problem in any deep or profound way except for the material contained in this section. By setting the authentication problem aside, significant progress can be made elsewhere.

1.3.7 Non-Technical Factors

In addition to technical issues, there may have been a variety of non-technical factors that have hampered the deployment of systems that are both secure and usable. Such factors could include concerns regarding liability, "turf-wars" and political battles within organizations, and the existence of organizations that benefited from the current state of affairs.

This thesis takes a broad view of how regulatory issues have traditionally affected the interaction of usability and security. It also finds that some traditionally difficult technical problems might be solvable through the use of relatively modest regulatory mechanisms that have had considerable success in other domains.

1.4 Why Have Usability Specialists Failed to Address Security Issues?

The previous section discussed some of the systemic reasons why traditional research on computer security issues has frequently ignored the issue of usability. This section addresses the problem from the opposite direction, and explores why usability researchers have generally ignored issues of computer security.

Although research on computer security goes back for decades, research in the field known as Human Computer Interactions (HCI) is much more nascent. Section 2.1.3 traces the emergence of HCI as a field from practitioners who were working in the field of "Social and Behavioral Computing" in the 1960s. A careful reading of that section—and the literature of the field in general—will reveal that issues involving computer security have received relatively little treatment over the past four

What Is Usability?

Usability is the measure of the quality of a user's experience when interacting with a product or system—whether a web site, a software application, mobile technology, or any user-operated device.

Usability is a combination of factors that affect the user's experience with the product or system, including:

Ease of learning How fast can a user who has never seen the user interface before learn it sufficiently well to accomplish basic tasks?

Efficiency of use Once an experienced user has learned to use the system, how fast can he or she accomplish tasks?

Memorability If a user has used the system before, can he or she remember enough to use it effectively the next time or does the user have to start over again learning everything?

Error frequency and severity How often do users make errors while using the system, how serious are these errors, and how do users recover from these errors?

Subjective satisfaction How much does the user like using the system?

Figure 1-3: The definition of usability promoted by the US Department of Health and Human Services, [US 04] based on [Nie93b]

decades. This section proposes several hypotheses as to why this might be the case.

1.4.1 A definition of "usability"

Before addressing the question "why have usability researchers largely ignored the issue of computer security," it is useful to arrive at a definition for "usability."

Although many people use an informal and personal definition of "usability"—software is usable if they can use it—a variety of more specific definitions of usability are available. For example:

- The US Government has adopted a formal definition of usability (Figure 1-3) based on [Nie93b] which measures usability according to five measurable quantities.
- Apple's documentation takes a holistic approach, which views application usability as encompassing everything from the consistent use of Macintosh technology to having applications that are fast, responsive, and reliable.
- Whitten and Tygar propose a definition for usable "security software" that software can satisfy if users understand the software's tasks, if they can figure out how to use the software to perform those tasks, if users don't make dangerous errors and if they are sufficiently comfortable with the software to continue using it (see [WT99] and Figure 2-5).

.

1.4.2 Historical disinterest in security

Much of the early work on usability simply ignored security issues, even when security was clearly part of the overall problem. In his paper "Iterative User-Interface Design,"[Nie93a] Nielsen presents the results of four usability studies, three of which have security functions in a central role. Yet each time there is an opportunity for Nielsen to comment on security issues, he avoids them:

- Nielsen attempts to establish the cost of user errors. “For example, an error in the use of a print command causing a file to be printed on the wrong printer has a cost approximately corresponding to the time needed for the user to discover the error and walk to the wrong printer to retrieve the output.”[Nie93a] But if the printout contains confidential information—for example, an employee offer letter with salary information—the cost associated with that information’s disclosure can be significantly greater than the minor inconvenience of having to walk to another printer.
- Nielsen discusses the examination of a “Home Banking” system under development. But while Nielsen considers the speed of consumers using the system and the chance of making an error, he fails to examine how users logged in, how they were authenticated, how passwords would be reset, or how resistant the system would be to so-called “phishing” attacks.[2]
- Nielsen’s study of a “cash register” application examined operator errors and speed, but did not examine the effectiveness of the authentication procedure. The “Security” application examined the speed of users authenticating with a single-sign-on system for a mainframe computer, but (at least his published report) didn’t examine issues such as spoofing, trusted path, or account lockouts.

Perhaps the reason that Nielsen and others ignored addressing security issues is that there was no need to do so. Sasse, for example, claims that most HCI-SEC problems are really conventional usability problems and can be solved most of the time using conventional usability approaches. She argues that many of the apparently insurmountable problems in HCI-SEC can be solved with user-centered design principles. [Sas04b, Sas03]

For example, Sasse and Brostoff [BS03] show that the seemingly insurmountable problem of password memorization can be dramatically mitigated by allowing users *ten* incorrect password attempts before performing account lockout, rather than the traditional *three*. The pair assert that the increase from three to ten tries does not significantly reduce security if strong passwords are employed.

1.4.3 Usability researchers were busy

One simple explanation for the lack of usability research addressing issues of computer security is that the field of usability emerged in the 1980s and 1990s, and that researchers during this time were busy exploring more fundamental usability questions, such as the appropriate way to make use of graphical input and output devices, the potential for handheld computing, and effective means for accessing the large amount of information that could be placed on CD-ROMs.

Yet another explanation is that security research was not a priority in the pre-networked world of

[2]Although phishing attacks seem to be a recent occurance, attacks of this type were experienced at MIT in the 1980s, when individuals posting as system operators sent email to computer users at the MIT Media Lab asking that the users change their password to a specific word or else e-mail their password back to the attacker. The first automated phishing system was the program AOHell, released in November 1994 by the hacker “Da Chronic.”[Gar95, Chr95, Raj03] Among its other capabilities, AOHell included an automated “fisher” designed to assist in stealing credit card numbers and other account information from other AOL users.

1980s and early 1990s because most computer systems were protected through physical security. This explanation holds that it was only after the mass popularization of the Internet that the usability of security systems became a serious societal issue. Before it was necessary for people to manage their own security in a networked environment, it was acceptable for security systems to be complex and difficult.

1.4.4 Psychological basis

It is also possible to suggest psychological explanations for the traditional disinterest in security issues by usability researchers.

While most usability researchers have devoted their professional careers to making computers easier-to-use, a sizable amount of computer security is devoted to making computers difficult for attackers to use. If there were no attackers there would be no need for passwords, cryptography, or antivirus systems. (Of course, there would still be need for backups and systems to protect against natural disasters, but the threat level would be significantly reduced.) Given that usability researchers want to be *enablers*, not *barriers*, it is entirely understandable that they would want to avoid a part of the computer discipline that would require them to make computers harder-to-use (at least, for the attackers).

Another psychological explanation is that usability researchers avoided security work because they did not view the work as being important. It is well known that people exaggerate minor risks while minimizing risks that are large but infrequent. Many usability researchers might have concluded that security threats were over-hyped and, in fact, less important to solve than other usability challenges.

Some usability researchers have further claimed that security researchers have been generally uninterested in usability issues—or even about users in general. Given this attitude, why would any self-respecting usability research want to work with a security professional?

1.5 Security Principles

The work presented in this thesis is based on six guiding principles for the aligning of security and usability. These principles, in turn, are based on an in-depth review of more than 30 years of computer security academic literature, face-to-face interviews (both formal and informal) with hundreds of computer security practitioners, and roughly two decades' experience working computer security in both academia and industry. Although this thesis does not provide guidelines for choosing security principles, the principles themselves provide guidelines for finding good patterns that align security and usability.

Chapter 10 presents the six principles for aligning security and usability:

- **The Principle of Least Surprise.** This principle is a restatement of Saltzer and Schoreder's principle of "psychological acceptability."[SS75] This principle holds that the computer should not surprise the user when the user expects the computer to behave in a manner that is secure. The Principle of Least Surprise is violated when there is a mismatch between the user's expectations and the computer's implementation. One way to correct such a violation is to educate the user; a second approach is to change the underlying computer system so that security properties mesh naturally with user expectations. It is observed that many security professionals spend the first decade of their career pursuing the first approach of educating users, and the rest of their career pursuing the second.

- **The Principle of Good Security Now.** Computer security is an engineering discipline. Even though it is impossible to have a computer system that is completely secure, there is always a tension between deploying good systems that are available today and waiting for better systems that can be deployed tomorrow. This principle holds that it is a mistake not to deploy good systems that are available now: if good systems are not deployed, end-users who are not trained in security will create their own, poor security solutions.
- **Provide Standardized Security Policies.** Today's security subsystems provide too many choices and configuration options that are relevant to security. These choices are frequently overwhelming to end-users. Worse, relatively minor changes in a security policy or configuration can have a drastic impact on overall security. Most users need security experts to make decisions for them because—by definition—users are not experts.

  This is not to say that users need to be locked in tight to a few inflexible policies from which they can never deviate. What's needed is a range of well-vetted, understood and teachable policies, and then the ability to make understood, controlled, contained and auditable deviations from these policies when needed.
- **Consistent Meaningful Vocabulary.** Usability is promoted when information is presented with a vocabulary that is consistent and meaningful. But, as will be shown in Section 8.2, there is a natural tendency among computer engineers to be loose with their choice of language. A guiding principle for aligning security and usability is that security information, at least, must be standardized and used consistently.
- **Consistent Controls and Placement.** In addition to standardizing vocabulary, it is important that security-related controls in graphical user interfaces be likewise standardized, so that similar functionality is presented in a similar manner and in a consistent location in user interfaces.
- **No External Burden.** Security tools must not pose a burden on non-users who do not otherwise benefit from their use. Otherwise, non-users will push back on users through social channels and encourage the users to discontinue the use of the tools.

These principles must be adapted with reason to the tasks that are at hand. Grudin observed that there are *many* cases in which a simple application of consistent user interface rules does not lead to interfaces that are easy-to-use.[Gru89] Instead, he argues that consistency with simplistic rules must often be violated in the interest of creating a user experience that it itself natural and consistent.

1.6 Original Contributions

This thesis summarizes a significant body of research performed at MIT over the past three years. Original contributions contained herein include:

General principles and literature review:

- A novel psychological and professional hypothesis for the traditional lack of work towards the goal of aligning usability and security.
- One of the most comprehensive reviews to date of the literature in the field of usability and security (excluding the literature that specifically addresses the issue of user authentication).
- A detailed analysis and critique of the definitions, principles, and findings put forth by Whitten and Tygar in their widely cited paper, *Why Johnny Can't Encrypt*,[WT99] and elaborated in Whitten's PhD thesis.[Whi04a]

On the topic of sanitization:

- An analysis of 236 hard drives acquired on the secondary market to determine the amount of confidential data left on the drives by their previous owners. (The *Remembrance of Data Passed* study.)
- Tools and techniques for automatically classifying information in hard drive images.
- A study based on interviews with individuals identified from the 236 hard drives to determine the individual or organizational failure that resulted in confidential information leaving the individual or organization's security perimeter. (The *Trace Back* study.)
- A study of how different operating systems handle file sanitization issues, and a proposal for a new operating system service called a "Shredder" for solving the file sanitization problem in a fashion than is both secure and usable.
- A study of how different web browsers leak personal information through improper sanitization of browser resources, and a proposal to overcome this problem through the unification of the browser cache and history facilities.
- A study of how improper sanitization in Microsoft Word and Adobe Illustrator documents has similarity resulted in the leakage of confidential information.

On the subject of PKI and secure messaging:

- A survey of 469 Amazon.com merchants regarding their attitudes towards and use of mail that is secured with cryptography.
- A novel technique for embedding digital signatures in Internet-standard email messages in a manner that is invisible to MIME-compatible email readers that do not know how to verify the signature.
- A statement and analysis of the Key Continuity Management (KCM) model for public key certification.
- A technique for adapting S/MIME-compatible mailers to work with KCM.
- A user study of Outlook Express with KCM that contrasts this model with the PGP key-signing model using the same protocol created by
- A meta-analysis of the E-Soft Secure Server Space for the period September 1998 through September 2004, showing that self-signed certificates have steadily increased in popularity over that time period.

Regulatory approaches for aligning security and usability:

- A “Bill of Rights” for the labeling and use of Radio Frequency Identification technology on consumer products.
- A proposal for a technique that would make hidden features of software could be made visible to computer users in a consistent manner through the use of visual warnings (icons).
- A novel analysis of how the ANSI Z535.4-2002 standard for Product Safety Signs and Labels could be applied to software products.

Other approaches for aligning security and usability:

- An analysis of how the inconsistent use of vocabulary in computer security contributes to usability problems, and an explanation as to why the use of inconsistent vocabulary is more likely in software than in other engineering fields.

Finally, this thesis introduces a set of principles and patterns that have the goal of aligning usability and security. This list, while not exhaustive, is designed to be something that can be given to an engineering team and readily applied to existing and next-generation computer systems and applications. The list includes:

—General Principles—

- **Least Surprise / Least Astonishment**: *Ensure that the system acts in accordance with the user’s expectations.*
- **Good Security Now (Don’t Wait for Perfect)**: *Ensure that systems offering some security features are deployed now, rather than leaving these systems sitting on the shelf while researchers try to develop “perfect” security systems for deployment later.*
- **Provide Standardized Security Policies (No Policy Kit)**: *Provide a small number of standardized security configurations that can be audited, documented, and taught to users.*
- **Consistent Meaningful Vocabulary**: *Prevent confusion by using words consistently to convey the same idea or concept in different programs and contexts. Likewise, prevent confusion by assigning consistent meanings to the same word in different applications or contexts.*
- **Consistent Controls and Placement**: *Structure applications so that similar functionality is located in similar positions on different applications—especially if those applications are manufactured by competitors.*
- **No External Burden**: *Design security systems to have minimal or no negative impact on the friends, associates and co-workers of those using the technology, so that they do not push back on the users of the tools and ask that the use be curtailed.*

—User Visibility and Sanitization Patterns—

- **Explicit User Audit**: *Allow the user to inspect all user-generated information stored in the system to see if information is present and verify that it is accurate. There should be no hidden data.*
- **Explicit Item Delete**: *Give the user a way to delete what is shown, where it is shown.*

- **Reset to Installation**: *Provide a means for removing* all *personal or private information associated with an application or operating system in a single, confirmed, and ideally delayed operation.*
- **Complete Delete**: *Ensure that when the user deletes the visible representation of something, the hidden representations are deleted as well.*
- **Delayed Unrecoverable Action**: *Give users a chance to change their minds after executing an unrecoverable action.*

—Identification and Key Management Patterns—

- **Leverage Existing Identification**: *Use existing identification schemes, rather than trying to create new ones.*
- **Email-Based Identification and Authentication**: *Use the ability to receive mail at a predetermined email address to establish one's identity or authorization to modify account parameters.*
- **Send S/MIME-Signed Email**: *Send email signed with S/MIME signatures to increase confidence in email, allow recipients to detect mail with forged* `From:` *headers, increase familarity with secure email through causal exposure and the resulting"passive learning," and give web-mail providers incentive to support S/MIME.*
- **Create Keys When Needed**: *Ensure that cryptographic protocols that can use keys will have access to keys, even if those keys were not signed by the private key of a well-known Certificate Authority.*
- **Key Continuity Management**: *Use digital certificates that are self-signed or signed by unknown CAs for some purpose that furthers secure usability, rather than ignoring them entirely. This, in turns, makes possible the use of automatically created self-signed certificates created by individuals or organizations that are unable or unwilling to obtain certificates from well-known Certification Authorities.*
- **Track Received Keys**: *Make it possible for the user to know if this is the first time that a key has been received, if the key has been used just a few times, or if it is used frequently.*
- **Track Recipients**: *Ensure that cryptographically protected email can be appropriately processed by the intended recipient.*
- **Migrate and Backup Keys**: *Prevent users from losing their valuable secret keys.*
- **Distinguish Internal Senders**: *Allow users to readily distinguish between mail that was generated from within an email system and mail that was injected from the outside but which claims to have an internal address.*

—Patterns for Promoting Overall Secure Operation—

- **Create a Security Lexicon**: *Provide a single location where security-releated words are defined, allowing the use of these words to be standardized within and between systems. The single lexicon should be consistent across vendors as well.*
- **Disclose Significant Deviations**: *Inform the user when an object (software or physical) is likely to behave in a manner that is significantly different than expected. Ideally the disclosure should be made by the object's creator.*

- **Install Before Execute**: *Ensure that programs cannot run unless they have been properly installed.*
- **Distinguish Between Run and Open**: *Distinguish the act of* running *a program from the opening of a data file.*
- **Disable by Default**: *Ensure that systems does not enable services, servers, and other significant but potentially surprising and security-relevant functionality unless there is a need to do so.*
- **Warn When Unsafe**: *Periodically warn of unsafe configurations or actions.*
- **Distinguish Security Levels**: *Give the user a simple way to distinguish between similar operations that are more-secure and less-secure. The visual indications should be consistent across products, packages and vendors.*

1.7 Thesis Roadmap

This thesis contains 11 Chapters and five Appendices.

Chapter 2: Prior Work
A comprehensive review of the literature in the field of usability and security, with special attention to theories of HCI-SEC, a survey of existing HCI-SEC techniques, and a discussion of regulatory solutions to the problem of unknown functions in programs and physical objects.

Chapter 3: Sanitization and Visibility 1: Operating Systems
An in-depth analysis of the data remanence problem—that is, data that is left behind on hard drives after it is no longer needed and/or intentionally deleted. This chapter presents the results of the "Remembrance of Data Passed" study in which 236 hard drives were purchased on the secondary market and analyzed to determine the information that had been left behind. Next this chapter presents the results of the "Traceback" study in which some of the original data owners were contacted to determine the reasons for the release of their confidential information. Finally, this chapter considers changes to the operating system that would overcome the data remanence problem.

Chapter 4: Sanitization and Visibility 2: Applications
This chapter shows how the patterns developed in Chapter 1 directly apply to other areas in which confidential information has been compromised. This chapter considers the release of personal information in web browsers and the release of deleted information in the Microsoft Word and Adobe Acrobat file formats. These inadvertent releases can be overcome through the use of the patterns put forth in the previous chapter.

Chapter 5: Solving Secure Email's "Grand Challenge" with Signature-Only Email
This chapter develops an argument that the use of digitally signed mail is a reasonable stepping

stone to the greater use of email security technology in general. The chapter begins with a review of the 28-year-history of secure email technology. Next presented are results of a survey of Amazon.com merchants, one quarter of whom had been receiving digitally signed VAT invoices for approximately one year. The survey found that these merchants believe that digital signatures are appropriate for the very kinds of email messages that they are sending and receiving. The survey also found that the merchants had no usability burdens in the receipt of mail that was digitally signed. Finally, the chapter examines different failure modes for digitally signed mail on today's desktop systems and discusses ways that both the programs and the signature standards that they implement could be improved.

Chapter 6: The Key Certification Problem: Rethinking PKI
This chapter examines what has been the primary stumbling block to the widespread use of secure email technology: the perceived need to certify public keys with the legally determined identities of the keyholders. It argues that this approach, put forth in the very first paper on public key technology[DH76], may not be an achievable goal, and in any case is not needed for the deployment and use of secure email technology.

Chapter 7: Johnny 2: A User Study of the Key Continuity Model
This chapter puts forth an alternative strategy for certifying public keys: Key Continuity Management (KCM). Next this chapter presents the findings of *Johnny 2*, a user study designed to test the viability of KCM with naïve users. *Johnny 2* replicates the scenario of Alma Whitten's *Why Johnny Can't Encrypt*[WT99], but introduces attacks on the test subjects. The study finds that Key Continuity Management offers effective protection against some but not all kinds of spoofing attacks.

Chapter 8: Regulatory Approaches
This chapter explores non-technical approaches for aligning security and usability—specifically the use of law and regulation to establish mandatory labeling regimes. These regimes are to expose hidden features of products and programs. Analogies are drawn from the 100-year history of labeling foods and drugs.

Chapter 9: Additional Techniques for Aligning Security and Usability
The previous four chapters looked in detail at a variety of design techniques for aligning security and usability. This chapter briefly considers some other approaches that appear promising and which support the design patterns presented in Chapter 10.

Chapter 10: Design Principles and Patterns for Aligning Security and Usability
This chapter formally presents the design patterns for aligning security and usability. Each pattern is presented in a stylized format that includes a specified *Pattern name*, *Intent*, *Motivation*, *Image*, *Applicability*, *Participants* (both other patterns and human actors), *Implementation*, *Results* and *Known Uses*.

Chapter 11: Future Work: An HCI-SEC Research Agenda
This concluding chapter suggests areas for future work in the short and long term. Finally, it suggests a number of preliminary patterns that could be further developed.

Appendix A: Hard Drive Study Details
This appendix presents details of the hard drive study presented in Chapter 3.

Appendix B: Mail Security Survey Details
Additional information and findings from the study of 469 Amazon.com merchants.

Appendix C: Johnny 2 User Test Details
This appendix presents technical details of the Johnny 2 user test.

Appendix D: Two Email Proxies
This appendix provides technical details of two proxies that were designed and implemented for this dissertation: Stream and CoPilot.

Appendix E: Specific Recommendations to Vendors
This appendix presents specific recommendations to Apple, Microsoft, and the Mozilla Foundation based on the work in this thesis.

Bibliography, Referenced Authors, Colophon Following the Appendices are the thesis references, a listing of references by last name, and a colophon that describes the book production with pdfLaTeX.