AUTHENTICATION AND SECURING PERSONAL INFORMATION IN AN UNTRUSTED INTERNET

by

Mohammad Mannan

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

School of Computer Science

at

CARLETON UNIVERSITY

Ottawa, Ontario
April, 2009

Table of Contents

List of Tables

List of Figures

Abstract

A large number of user PCs are currently infected with different types of malicious software including spyware, keyloggers, and rootkits. In general, any Internet-connected end-host cannot be fully trusted. In addition to this *compromised host* problem, attacks exploiting usability drawbacks of web services and security tools when used by everyday users, and semantic attacks such as phishing are commonly observed. In the given untrusted environment, traditional threat models which assume trusted end-hosts need to be re-evaluated. We propose a number of techniques to improve the trustworthiness of the web considering the current untrusted environment.

To understand what is expected from regular users for performing sensitive online tasks, we review security requirements of six Canadian online banks, and identified an emerging gap between these requirements and usability. Instead of requiring users to follow an extensive list of security *best-practices* for online banking, we propose the Mobile Password Authentication (MP-Auth) protocol. Using a trusted personal device (e.g., cellphone) in conjunction with a PC, MP-Auth protects a user's long-term login credentials, and offers transaction integrity assuming the user PC is untrustworthy and the user is unaware of phishing attacks. MP-Auth's security largely depends on user-chosen passwords, which are generally *weak*. To assist users in generating strong but usable passwords, we propose an Object-based Password (ObPwd) scheme which creates text passwords from user-selected objects, e.g., photos or music files.

As part of the compromised host problem, we further assume that sensitive identity numbers (e.g., Social Insurance Number) will eventually be breached. To reduce the value of compromised credential information to attackers in such a scenario, we propose the use of *localized* ID numbers that are valid only for a particular relying party. A similar localization approach for banking PINs to prevent exploitation of compromised PINs from intermediate banking switches is also proposed.

Acknowledgements

When I was working as a software developer in Vancouver in 2003, I had a phone interview with Prof. Paul Van Oorschot – as part of exploring the possibility of becoming his graduate student. During that discussion Paul raised the following simple and apparently trivial question: “How do you verify the authenticity of a website?” It was this question, and my appreciation for the *beauty* of public key cryptography (which I got exposed to as part of an undergraduate course), that brought me into the exciting field of security research with the focus on improving real-world security. I thank Paul for introducing me to this field which can significantly improve everyday people’s lives. I gratefully acknowledge his prompt, insightful, clear, and frank feedback; his foresight on identifying values, shortcomings, and obstacles in a particular proposal; his aptitude for putting an idea into context; his extra-ordinary eyesight for details; for always being positive and reminding me of the bright side of life especially in difficult times; and above everything, for being patient and helping make the long process of this work a meaningful and worthwhile endeavour. I literally cannot thank him enough.

The Carleton Computer Security Lab (CCSL) played a very significant role in my life for the last few years. CCSL members – my dear friends and colleagues – positively critiqued and supported me. Almost every part of this thesis has been presented at CCSL meetings, and discussed/debated extensively. They helped me sort out the better ideas from many not-so-worthwhile ones, to say the least. Several members graciously spent their time in reviewing early drafts of my work (including hundreds of lines of program code); even in very busy times, they managed to read and share their insights on my preliminary drafts. Many short walks to Tim Hortons, extended discussions at late hours, and long trips to conferences and workshops with CCSL members significantly shaped the outcome of this thesis. It was a great honour to work with so many bright minds. I thank all CCSL members from 2003 to 2009, especially Glenn Wurster, Julie Thorpe, David Whyte, Abdulrahman Hijazi, David Barrera, Mansour Alsaleh, Tim Furlong, Deholo Nali, James Muir, and Hajime Inoue.

I would like to express my sincere gratitude to the members of the thesis committee: Prof. Matt Bishop for agreeing to be the external examiner; Prof. Liam Peyton for his detailed comments; Prof. Robert Biddle for his insights on usability issues; and Prof. Anil Somayaji for his help in shaping our threat models.

Most parts of this thesis have been peer-reviewed. I would like to acknowledge the insightful feedback, and constructive suggestions from many anonymous reviewers. Special thanks to Brian Snow for shepherding the ID localization proposal; Bryan Parno (CMU) for allowing access to source code of his Phoolproof implementation, Anupam Datta (CMU) for helping with the PCL analysis, and Masud Khan for providing a Nokia E62 smartphone; Bernhard Esslinger of University of Siegen, and Joerg-Cornelius Schneider and Henrik Koy of Deutsche Bank, especially regarding attacks on the simple version of salted-PIN; and a reviewer from a large Canadian bank. Comments and suggestions from attendees at different conferences/workshops helped in the evaluation of the proposals from varying perspectives.

I received substantial financial support from several sources including: NSERC Canada Graduate Scholarship (CGS-D3), Public Safety and Emergency Preparedness Canada (PSEPC), Carleton University, and NSERC ISSNet. I am thankful to all for easing the financial burden while doing this research.

Many people helped me prepare for this long academic journey. Special thanks to my undergrad thesis supervisor Prof. M. Kaykobad (BUET, Bangladesh) for helping me understand the value of academic research; Partha Pande (WSU, Pullman) for consistently promoting the importance of graduate research; and my family members, especially my brother Anowar Hossain, for always inspiring me to aim for the best.

Being the spouse of a grad student is not easy. Thanks to my wife Ayesha Siddiqa for coping with my unpredictable schedules; for understanding the challenges in a grad student's life; for being the first one to hear my 'innovative' ideas; and for always supporting me in many ups and downs in the past several years.

Chapter 1

Introduction

In this chapter, we discuss overall motivation for this thesis and summarize our contributions.

1.1 Motivation

Since the early days of computer security, the intermediate network between two end hosts has been treated as the primary point of compromise. Security protocols are generally designed with a strong *network adversary* in mind who can learn, drop, modify any messages from the network (known as the Dolev-Yao threat model [69, 179]). The widely used SSL protocol is a prime example of this principle – the endpoints are trusted (i.e., malware-free), and the devil is in the network. While such assumptions may have been true in the early days of networked computers, the evolution of the Internet has resulted in a different reality. Today an adversary generally lives closer to end hosts than the intermediate network. This is in fact true for both ends of a typical communication channel: client PCs harbour numerous forms of malware, to such extent that apparently the computer industry and security research community have largely failed at the desktop security layer. On the other end, more and more service providers' systems (e.g., corporate, government) are also being compromised by organized hackers, exposing millions of user records with sensitive identity and financial information. Traditionally the primary focus of most computer security researchers has not been threats that plague the current Internet environment, such as keyloggers, rootkits, and phishing, although signs of awareness are slowly emerging [146]. For example, common threats to cryptographic protocols, as summarized by Boyd and Mathuria [44, pp. 23–31], only include eavesdropping, modification, replay, preplay, reflection, denial of service, typing attacks, cryptanalysis, certificate manipulation, and protocol interaction.

Apparently, the battle of saving Internet-connected general purpose consumer PCs against malware has been lost with millions of PCs in unhealthy states. For example, according to different estimations (e.g., [126]), the *botnet* as created by the Storm Worm, consisted of one to fifty million compromised PCs worldwide.[1] Many theoretical proposals and existing products that are designed to improve information security largely fall apart when we consider the current compromised state of Internet-enabled computers. We call this the "compromised host problem."

Additionally, semantic attacks such as phishing/web-spoofing are also rampant in the web world; these attacks largely exploit non-expert everyday users' *mental models* regarding the current web technologies and security products. Usability barriers as revealed in several studies (e.g., [67]) may severely reduce the effectiveness of many otherwise well-designed security tools. Security researchers of today are facing an extraordinary situation: we have PCs that are not trustworthy, which are being operated by everyday non-expert users, who are expected to use security tools that are *implicitly* designed for expert users. We believe that the current web is untrustworthy for several fundamental reasons including compromised end-hosts and the naivety of common online users, and that it is hard to make progress without directly addressing these fundamental issues.

Given such a scenario, we argue that many previously accepted techniques for information security should be re-evaluated to measure their efficacy in the present environment. We believe that counter-measures protecting against these current threats must be embedded into many existing security mechanisms if the goal is to provide effective security in practice. In this thesis, we propose a number of example techniques and protocols to improve trustworthiness of the current web environment. We also argue that one reason for the severity of current identity fraud, spam, phishing, and many other Internet-related attacks is the leverage gained by using data compromised from one site at many others, repeated times. This "compromise once, reuse multiple times" feature provides significant advantages to attackers. Part of this thesis aims to reduce this asymmetric leverage attackers currently enjoy.

[1] Estimating the size of a botnet remains elusive in practice; see e.g., [207].

1.2 Thesis Statement

Following the above discussion, the primary thesis or doctrine which runs throughout and motivates our work is summarized in three assertions.

Assertion 1. Many academic threat models as well as commercial security tools that are designed to improve security of Internet-enabled consumer PCs, need *updating* to reflect the current reality, e.g., to recognize the compromised host problem and the prevalence of semantic attacks against users.

Assertion 2. Usability must be incorporated into security designs, if the security problem we are addressing is related to consumer PCs, since past security models were mostly built for technical experts, whereas many of today's technologies are being used by total non-experts (everyday people, most with zero formal computer training).

Assertion 3. The asymmetric advantage of "compromise once, reuse multiple times" as currently exploited by attackers must be addressed to make the job of defenders of sensitive information easier and more manageable than the current *arms-race.*

The goals of our research are twofold: (i) to better understand the current environment by studying proposed and available security mechanisms for everyday non-expert users; and (ii) to explore specific problem instances which represent the above assertions, and propose new solutions to these problems. The hope is that by doing so, we may gain a better understanding of the broad classes these problem instances fall into, and potentially propose broader solutions or guidelines for improving Internet security mechanisms used by non-specialists. The main research questions we would like to explore in this thesis are:

- ***Question 1:*** Can we design instances of technologies that can improve security and privacy in real-world applications, given the current state of compromised computing environment?

- ***Question 2:*** What would be the design criteria of such solutions? Can we suggest any general guidelines?

1.3 Main Contributions

In this section, we briefly outline our contributions in three areas: (i) security-usability chasm in practice; (ii) password authentication and integrity verification; and (iii) design for damage control.

1.3.1 Security-Usability Gap

Most security tools were designed by security experts for *expert* users. We explore the gap between security and usability when security tools are used by non-expert, everyday users.

c1: The emerging gap between security and usability. To gain insight on what skills are expected from regular home users to perform sensitive tasks over the Internet, we review security requirements of Canadian online banks. In one sense, these requirements and expectations of user abilities give a snapshot of the current state of user-centric security issues, and help us to distinguish and analyze the major security tools that are expected to be used by non-expert users for doing business in a malware-infested environment. We identify an emerging gap between these requirements and usability from an analysis of online banking requirements as found in formal (legal) client agreements for regular and electronic banking, and security and privacy requirements (and recommendations) from banks' websites. From a small-scale questionnaire-based survey, we found that even security-aware users fail to satisfy most common online banking requirements such as password renewal, using anti-malware tools, and maintaining an up-to-date (patched) operating system and browser. Our work provides a glimpse into overall system security of home computers owned/operated by technically advanced users. Security attributes and user habits as revealed in our study provide a view to what security and usability researchers may expect at best from average users; we believe this is essential for informed discussion and design for usable security. We use insights from this study in designing security protocols and techniques as presented in this thesis.

1.3.2 Password Authentication and Integrity Verification

We list three major contributions in this area including two authentication protocols and a password generation technique. A set of non-cryptographic mechanisms for improving transaction integrity is listed as a minor contribution.

c2: Authentication technique for tasks involving sensitive information. We argue that in the current environment, especially due to phishing and *session-hijacking* attacks,[2] even when the current "security best practices" are met, there is no guarantee that a user would enjoy a safe online environment. Instead of requiring users to maintain a *healthy* (malware-free) PC and be able to detect complex phishing scams, we design a technique that will largely protect a user's security even when the user performs sensitive Internet tasks from a compromised PC, and is uninformed regarding semantic attacks.

As part of addressing the compromised end-user machine problem with regard to safeguarding sensitive long-term passwords (e.g., those used for online banking), we build on the following simple idea: use a hand-held personal device, e.g., a cellphone or PDA to encrypt the password (combined with a server-generated random challenge) under the public key of an intended server, and relay through a (possibly untrusted) PC only the encrypted result in order to login to the server website. This simple challenge-response effectively turns a user's long-term password into a one-time password in such a way that long-term passwords are not revealed to phishing websites, nor to keyloggers on the untrusted PC. The resulting protocol, called *MP-Auth* (short for *M*obile *P*assword *Auth*entication), also protects against session hijacking, by providing transaction integrity through a transaction confirmation step. Unlike standard two-factor techniques, MP-Auth does not store any secret on the mobile device.

c3: Authentication technique for better privacy of personal content. Arguably, most online applications are not as critical as online banking; thus MP-Auth's strong security features may be viewed as excessive for certain web applications such

[2] In a session hijacking attack, typically hidden malware on a client machine can access an established user session, change transaction details (e.g., charging $500 for a $5 item), or perform any unwanted operations, according to silent instructions from a remote attacker.

as authentication for sharing personal content on the web. Usability issues, such as difficulties associated with managing yet more shared passwords for authenticated personal web content distribution, cause most publishing users put their content online without any access restrictions. In some cases, this universal access adversely affects a web publisher's privacy. Leveraging *the circle of trust* in the existing contact list feature of Instant Messaging (IM) networks, we propose a protocol for privacy-enhanced personal content sharing called IM-based Privacy-Enhanced Content Sharing (*IMPECS*). It uses IM as a (transparent) distribution channel for authentication 'tickets' to viewing users as allowed by content publishers. IMPECS enables a publishing user's personal data to be accessible only to her IM contacts. A user can put her personal webpage on any web server she wants (vs. being restricted to any specific website, e.g., social networking sites), and maintain privacy of her content without requiring site-specific passwords. In our opinion, easy access to usable privacy tools may favourably change the actions of ordinary web users towards online privacy; IMPECS is designed to be such a tool to enhance privacy of personal web content.

c4: Mechanism for increasing entropy of user-chosen passwords. Several of our proposals (e.g., MP-Auth, IMPECS) rely on user-selected text passwords. As evident from different studies (e.g., [90]), motivating everyday users to choose 'strong' passwords has largely failed; for example, 'password1', 'abc123', and 'myspace1' were found to be the top three choices of passwords in one study [233]. To reduce apathy towards stronger passwords, we design a technique called Object-based Password (*ObPwd*) allowing users to generate strong text-based passwords from personally-meaningful digital objects of their choice. Objects such as multimedia files, images, and text snippets can be chosen from the web or users' personal media. Instead of remembering the generated password, users need to recall only the password generating object. We believe that recalling a *pointer* to such digital objects will be easier than following many password rules as advised/imposed by security experts.

c5: Integrity verification for financial transactions. While it is important and desirable to have authentication, confidentiality, integrity and privacy protections for every online/offline financial transaction, in reality, all these goals can be simultaneously achieved only rarely; this appears even more difficult under the current environmental attacks (e.g., malicious software, and semantic attacks such as phishing). Instead of aiming for 'ideal' security, we propose several mechanisms with the goal of simply verifying integrity of a given transaction, and detecting misuse of compromised credentials as early as possible. We argue that empowering users with information regarding the actual state of their accounts and transactions may help improve trustworthiness of online transactions. Our proposals combine and take advantage of existing techniques and academic proposals from the recent past. These techniques mitigate the effects of breaches which are difficult to prevent completely, e.g., a compromised password in MP-Auth.

1.3.3 Design for Damage Control

Assuming breaches of sensitive, long-term personal identification numbers are inevitable, we apply a *localization* technique (which need not be tied to a particular geographic or physical location) for reducing threats from such breaches.

c6: Localized identity numbers. From the endless reports of large-scale data breaches (e.g., [58, 26, 203]) exposing sensitive personal identity numbers of millions of consumers, we propose starting with a new assumption: sensitive user data *will* eventually be breached (cf. [146]). We further assume that this will occur primarily through "credential relying parties." We focus on how to nonetheless prevent identity fraud in such a scenario through the use of *localized ID numbers* that are tied to each particular relying party. An individual may be required to provide their SIN or driver's license number to many parties, e.g., employers, banks, credit reporting agencies, car rentals, all of whom often keep sensitive identification details for a long time. Confidentiality of such data may be breached by any of these parties. If a localized SIN scheme were in place, where each employer would get a "different (non-reusable) version" of the SIN of the given individual, then a disclosure of any such SIN would not be useful for identity fraud. We propose several techniques based on this idea.

c7: Localized/salted banking pin. Despite best efforts from security API designers, flaws are often found in widely deployed security APIs. In parallel to spending research efforts to improve the security of these APIs, we argue that it may be worthwhile to explore design criteria that would reduce the impact of an API exploit. We use such a design philosophy in dealing with PIN cracking attacks on financial PIN processing APIs. Our solution is called *salted-PIN*: a randomly generated salt value of adequate length (e.g., 128-bit) is stored on a bank card in plaintext, and in an encrypted form at a verification facility under a bank-chosen *salt key*. Instead of sending the regular user PIN, salted-PIN requires an ATM to generate a *Transport Final PIN* from a user PIN, account number, and the salt value (stored on the bank card) through, e.g., a pseudo-random function. We explore different attacks on this solution, and propose variants of salted-PIN that can protect against known attacks. In particular, our *localized salted-PIN* incorporates local information such as ATM location and identification into PIN generation. If such a localized PIN is compromised, it can only be used from a particular ATM/location, thus limiting the exploitation of such PINs.

1.4 Related Publications

Most parts of this thesis have been peer-reviewed. These publications are listed below in a chronological order.

1. Using a Personal Device to Strengthen Password Authentication from an Untrusted Computer. M. Mannan, P.C. van Oorschot. *Financial Cryptography and Data Security 2007 (FC'07)*, Feb. 12-15, 2007, Scarborough, Trinidad and Tobago.

2. Security and Usability: The Gap in Real-World Online Banking. M. Mannan, P.C. van Oorschot. *New Security Paradigms Workshop 2007 (NSPW'07)*, Sept. 18-21 2007, New Hampshire, USA.

3. Weighing Down "The Unbearable Lightness of PIN Cracking." (short paper) M. Mannan, P.C. van Oorschot. *Financial Cryptography and Data Security 2008 (FC'08)*, Jan. 28-31, 2008, Cozumel, Mexico.

4. Privacy-Enhanced Sharing of Personal Content on the Web. M. Mannan, P.C. van Oorschot. *World Wide Web Conference 2008 (WWW'08)*, Apr. 21-25, 2008, Beijing, China.
5. Digital Objects as Passwords. M. Mannan, P.C. van Oorschot. *USENIX Hot Topics in Security 2008 (HotSec'08)*, San Jose, California, USA, July 29, 2008.
6. Localization of Credential Information to Address Increasingly Inevitable Data Breaches. M. Mannan, P.C. van Oorschot. *New Security Paradigms Workshop 2008 (NSPW'08)*, Lake Tahoe, California, USA, Sept. 22-25, 2008.
7. Reducing Threats from Flawed Security APIs: The Banking PIN Case. M. Mannan, P.C. van Oorschot. *Elsevier Journal of Computers & Security* (in press; accepted on Jan. 26, 2009).

1.5 Organization

In Chapter 2, we analyze security requirements for online banking, and comment on several usability and security issues. An authentication mechanism to perform sensitive online tasks from a compromised PC, along with security and attack analysis, and implementation detail has been provided in Chapter 3; related work and complimentary techniques to this mechanism are discussed in Chapter 4. In Chapter 5, we introduce a protocol for privacy-enhanced sharing of online personal content. A technique for generating better text passwords (i.e., with more entropy than user-selected passwords) is discussed in Chapter 6. Techniques for restricting threats from breached personal identification numbers, and banking PINs are discussed in Chapters 7 and 8 respectively. In Chapter 9, we provide a comparative summary of threats addressed by our proposals, revisit the thesis objectives, and list the lessons learned from this thesis.

Chapter 2

On the Emerging Gap between Security and Usability

Sensitive online tasks such as online banking require users to maintain a (malware-free) *clean* computer, and to guard against semantic attacks such as phishing. In this chapter, we discuss our study on online banking security requirements as expected from regular users by several Canadian banks. One motivation behind this study is to shed light on the absurdity of these requirements in terms of usability, and to fathom severe limitations of these security measures even when fulfilled.

2.1 Introduction

Applications with major security and usability issues, such as online banking, are being used by more and more people who are less and less technically savvy. Most major banks currently support online banking, as it enables them to serve far more customers than by traditional banking, at a fraction of the cost.[1] Exploiting the convenience and overhead savings possible through the Internet, some online-only banks have also arisen. Online access also reduces physical visits to the bank, which saves customers' an hour and fifteen minutes (as advertised by one bank). However, the popularity of online banking has attracted criminals exploiting (online) banking customers. Attacks have been launched against customers of big and small banks worldwide. Canadian banks – some of which are among the largest in the world – are no exceptions. There are five major traditional banks operating in Canada which are quite heavily regulated by the Canadian Bankers Association and the government of Canada. These banks provide comprehensive guidelines to their online customers as part of the campaign against Internet-based attacks. The 2007 Canadian Internet Use Survey [250] found that 63% of Canadian Internet users do online banking. However, it was not studied how many users actually fulfill their online banking responsibilities as dictated by the

[1] One study [141] estimates in-branch transactions cost about $1 to $4 while an online transaction costs less than five cents.

banks. As a typical example, users are expected to install and maintain a firewall, anti-virus, and anti-spyware programs on their own. A user could be held responsible for financial losses if their PIN or password is “easily” guessed, e.g., derived from their phone number, birth date, address or social insurance number (SIN).

Globally, banks are becoming increasingly reluctant to reimburse users who fall prey to online scams such as phishing. We are not aware of any such reported cases in Canada so far although numerous such incidents have made news in the recent past elsewhere. One report [100] indicates that U.S. victims of phishing attacks lost five times more money in 2006 than 2005. Although 80% of the victims in 2005 got their money back, in 2006 only 54% victims were refunded by their banks. In the U.K., bank card related frauds are on the decline over the first six months of 2006, but online banking fraud increased by 55% during this period. Online banking fraud victims of China must prove that the bank was at fault to get any money back. According to the 2007 Code of Banking Practice in New Zealand [187], banks may inspect a victim’s computer in case of an Internet fraud to verify if the customer has met all the online banking security requirements (e.g., anti-spyware, operating system security updates). Netcraft news [182] reported that fraud victims of the Bank of Ireland were denied reimbursement when they lost about $160,000$ euros through phishing attacks. When threatened by lawsuits, the bank finally refunded the victims. To avoid further losses banks worldwide are imposing increasingly more responsibility on online banking users. The fear of losses is also having an impact on users. A survey of $23,000$ European Internet users reported [243] that security worries deter 40% of those surveyed from doing online banking. A Gartner survey [61] reported that online attacks influenced nearly 30% of online banking users; more than 75% of those users logged in less frequently, and about 14% stopped paying bills online.

In conjunction with this study, we opened personal checking accounts at the five largest Canadian banks: CIBC, RBC Royal Bank, TD Canada Trust, Scotiabank, and BMO Bank of Montreal.[2] We also opened an account at President’s Choice Financial, which is primarily an online-only bank. We opened these accounts during October and November of 2006, and have been accessing them up until April 2009. Our study

[2] Many of the 32 million Canadians have accounts at more than one of these.

of online banking security and usability is mostly a *cognitive walk-through* [277]. We focus mainly on issues related to personal checking accounts (vs. business, trading, or credit card accounts). We critique privacy and security requirements[3] imposed by banks on regular home-based online banking users from a usability point of view (i.e., whether these requirements are reasonable and practical). Most of our reported findings were derived from our on-site bank account creation experiences, content of bank agreements (for regular and electronic use), security and privacy requirements and recommendations from bank websites, and our experience of using the online banking interfaces. Our findings lead us to believe that most users fail to meet the requirements enabling eligibility for the 100% reimbursement guarantee for online banking fraud losses. We also conducted a questionnaire-based survey among technically advanced users to understand whether even they satisfy online banking requirements. The survey indicated that indeed, most participants failed to fulfill their banks' online banking requirements.

Putting our work in perspective, user-centered security [299] was introduced by Zurko and Simon at NSPW 1996 (see also [298]). Several user non-acceptance paradigms were explored in an NSPW 2004 panel [108]. Online banking is no longer *new*, but still relatively young in its life, and increasingly new converts to online banking are less and less technically savvy. Security and usability is still a relatively *new* paradigm from an academic research perspective. Our work provides a reality check on usable security, using online banking as an example of widely used security-sensitive applications.

Contributions/discussion points. We analyze banks' requirements for online banking from a usable security perspective, using Canadian banks as a case study. Our contributions and discussion points include:

1. Specific Online Banking Security and Usability Issues. We provide an analysis of online banking requirements from client agreements

[3] Requirements here include conditions from banks' customer agreements (hard-copy and/or electronic) and security recommendations/advice from bank websites. The legal implications of banks' recommendations and security guidelines are unclear.

(for regular and electronic banking), and security and privacy requirements/recommendations from banks' websites. These highlight real-world security and usability issues.

2. User Survey on Satisfying Online Banking Requirements. We report on a questionnaire-based survey of 123 computer science students, professors, researchers and professionals. Even most of these technically-advanced users fail to satisfy common online banking requirements in practice.
3. Glimpse of System Security for Informed Research. What kind of systems do people actually use for highly-sensitive online tasks such as Internet banking? Banks' security requirements for online banking (e.g., password, anti-malware, up-to-date operating system and browser) encompass several aspects of system security for average users. Our work provides a glimpse of overall system security of home computers owned/operated by technically advanced users. Security attributes and user habits as revealed in our study provide a view to what security (and usability) researchers may expect at best from average users; we believe this is essential for informed discussion and design for usable security.
4. Who Bears the Responsibility for Security. In the future, we may be using Internet-enabled home computers for tasks potentially even more security-sensitive than online banking. But even for online banking, is it enough for service providers to simply impose on users, in a customer agreement, whatever responsibilities they deem appropriate? On the other hand, should users take no responsibility for the security of their PCs and Internet usage? Where should the balance of responsibility rest? Our work aims to spur discussion on these questions.

Overview. Related work is discussed in Section 2.2. SSL-based bank site authentication is discussed in Section 2.3.1. Banks' anti-malware requirements are discussed in Section 2.3.2. We discuss banking agreements in Section 2.3.3. Software updates and miscellaneous issues (including user authentication) related to online banking are discussed in Sections 2.3.4 and 2.3.5 respectively. We provide the results of our online banking survey in Section 2.4. Section 2.5 concludes.

2.2 Related Work

Related work is discussed both here and other appropriate places throughout this chapter. Hertzum et al. [121] analyzed the usability of Danish online banking security for a particular task (money transfer to a specific account) with respect to the usable security definition of Whitten and Tygar [278]; and evaluated installation of special client-end e-banking software (as required by some Danish banks), logon, money transfer, and logoff. Several usability weaknesses in online money transfer were revealed. Chung et al. [52] studied the effectiveness of web interfaces of several banks in New Zealand. Nilsson et al. [188] exposed the security implications of system generated versus user-chosen password for online banking. Singh [245] examined (through a user study) the users' perspective on the security of online banking in Australia, and concluded that one way to increase users' perception of online security is "to have customers believe the provider will not allow them to suffer fraudulent transactions." Edge et al. [74] analyzed online banking (money transfer in particular) security using *attack trees*, and proposed solutions to known attacks using *protection trees* and multi-factor authentication. Karjaluoto et al. [138] conducted a survey among non-users and users of Finish online banking, and analyzed differences of the demographic profile of these two groups of users. A similar survey [31] attempts to examine the effects of trustworthiness among online banking users. Jin et al. [134] briefly analyzed online banking risks for banks, and how these risks may be managed. As opposed to analyzing the usability of offered features or any specific security mechanism (e.g., password) for online banking, we focus on the usability of major online banking requirements.

2.3 Requirements and Recommendations for Online Banking

In the following subsections, we discuss security requirements and recommendations related to SSL certificate, anti-malware, documentation, software updates, and authentication as appeared in banking sites and legal agreements (mostly from a period of Oct. 2006 to Nov. 2007).

2.3.1 Bank Site Authentication: SSL Certificate

In this section, we discuss what we see as serious usability issues related to SSL certificates. Bank websites generally present an SSL certificate on an online login page to be authenticated by a user. Banks expect users to visit the correct URL of a bank, check for visual clues (e.g., the lock icon and `https` on the address bar) of an SSL protected site, and check certain items on the site certificate before entering any login information. Banks emphasize that the presence of an SSL certificate implies that a website is *secure* and *genuine*, and that no one can see a user's information other than the bank (e.g., BMO states that an SSL certificate is an electronic *passport* for a website). Although a certificate may authenticate a website, it of course does not *secure* the site, and malware on a PC can easily access all (including SSL-protected) user information. Below we discuss more SSL-related issues in detail.

Login URL. Users may be redirected to spoofed or malicious websites if a memorized bank URL is misspelled. A bookmarked login URL may be replaced by a phishing site URL by malware on a user PC. Banks strongly recommend users to check the URL of a bank website before entering the bank card number and password. Nevertheless, when a login URL is `https://www.txn.banking.pcfinancial.ca/a/authentication/preSignOn.ams?referid=loginBox_banking_go`, it is unclear how users should make a decision as to whether this is a *correct* URL. Some banks (e.g., PC Financial) even state that the 's' in the `https` implies the website is "secure" even though a self-signed site certificate can easily be used to show `https`. In a user study as reported by Dhamija et al. [67], 23% of users relied only on the content of a page to determine legitimacy, i.e., did not check the URL at all, and many users did not understand the syntax of domain names. Downs et al. [70] reported that only 35% of the participants of a user study noticed the text `https` on a URL; some who noticed the `https` did not understand the significance of the extra 's.' Nearly 45% did not look into the URL at all in their study. We thus believe that banks have unrealistic expectations of users *checking* URLs.

The SSL lock icon. The display of an SSL padlock is strongly emphasized as an indication of a secure website (though not all emphasize that this should be on the browser chrome). As banks state, generally the lock icon is displayed on the lower-right corner of a browser. However, different browsers, and even different versions of the same browser, show the padlock in different places on the browser chrome. Internet Explorer 7 (IE7) has moved the padlock location from the traditional lower-right corner to the address bar. Although CIBC and PC Financial login pages have SSL certificates, the Opera browser (version 9.22 on Windows XP, and version 9.2 on Mac OS X) does not display the lock icon unless the login page is re-loaded (i.e., refreshed); in fact, Opera's site information dialog states that "Site not secure...The communication is done in plain text, and there is no way to guarantee the identity of the server." PC Financial states that there is a known bug affecting earlier versions of IE (e.g., 5.5); when a user selects the padlock icon, IE warns, "This certificate has failed to verify for all of its intended purposes." The bank advises users that this is not a concern as the SSL certificate is actually okay.

The lock icon is displayed inside the sign-on pages of all banks. Many phishing websites use a closed-padlock inside the browser chrome – which has nothing to do with SSL protection – to *assure* users that the website is secure. Average online users do not generally understand that any webpage can display within the page content itself whatever icon or text the page designer wishes; thus an embedded lock icon may conflict with the SSL lock icon (on a browser chrome) and confuse users. A website can also display any icon at the beginning of the site's URL as a *shortcut icon.* This icon is displayed on most current browsers' address bar. For example, the website `http://www.ljean.com` (as of Aug. 27, 2007) has the lock icon as its shortcut icon although this is not an SSL protected site.

Only a small portion of users (23%) has been reported [67] to look for or notice the padlock icon on the browser chrome. Recently an increasing number of phishing sites is reported to have SSL certificates (mostly self-signed) to gain confidence of somewhat technically advanced users [181], i.e., users who might check the existence of a closed-lock and the corresponding SSL certificate. Another user study by Downs et al. [70] reported that most participants (85%) noticed the secure site lock icon on a website. However, only 40% of those who were aware of the lock icon knew that the lock must be on the browser chrome to signify an SSL-enabled website. Some participants stated that SSL certificates were a “formality,” like an “elevator certificate.” 32% of users would log into a website even if the site presents a self-signed certificate. In one real-world incident [181], only one in 300 customers of a New Zealand bank chose to abandon the SSL session upon a browser warning indicating an expired SSL site certificate; the bank accidentally allowed a certificate to expire for a period of 12 hours. We conclude that banks have unrealistic user expectations with respect to the SSL lock icon.

Security toolbar. TD Canada Trust provides a free tool from Symantec called Norton Confidence Online (NCO).[4] NCO is installed as an ActiveX control in IE, and is designed to detect spoofed TD Canada Trust websites. The tool is added to IE toolbar, and is displayed as a closed lock, similar to the SSL lock. Such visual similarity may confuse average online users. A user is notified with a dialog box when a spoofed TD site is detected, but users may ignore this and visit the site anyway [189]. NCO is available only when accessed from IE on Windows. Installation of the ActiveX control requires administrative privileges. Also any such toolbars may still miss a significant portion of phishing websites as revealed by Zhang et al. [297] (similar results were reported by Wu et al. [287]). In fact, in one recent (Aug. 2007) real-world case [33], when the Bank of India website was compromised for several hours and serving more than 22 pieces of malware to each site visitor, most well-known online *trust brokers* (e.g., Netcraft toolbar, McAfee SiteAdvisor, Google Safe Browsing) reported the site to be valid.

Certificate components.[5] Banks ask users to check the SSL site certificate of a bank on a login page, but do not illustrate exactly what a user should look for in a certificate. Table 2.1 summarizes important components of SSL certificates provided on sign-on pages of different banks. As evident from the Common Name (CN) column of Table 2.1, CNs are quite arbitrary and different than banks’ advertised home-page URLs. For example, CIBC’s main webpage URL is `www.cibc.com`, but the CN on the sign-on page SSL certificate is `www.cibconline.cibc.com`. However, there is a different SSL certificate for CIBC’s ‘Contact Us’

[4] This tool is apparently unavailable as of Apr. 18, 2009.

[5] We do not discuss IE7 *extended validation* certificates, which solve some problems but arguably raise many others [131].

page with www.cibc.com as the CN. After logging into online banking, the TD SSL certificate changes (for no obvious reason), showing different names as CN, e.g., easyweb37z.tdcanadatrust.com, easyweb45z.tdcanadatrust.com. Following the tabs on the online baking page brings other TD pages with different CNs on their SSL certificates, e.g., www.tdcanadatrust.com, www.tdwaterhouse.com (this is true for other banks too). Understanding the details of an SSL certificate is already complicated for many users, and so many different SSL certificates may confuse users even further.

	Common Name (CN)	Organization (O)	Signing CA	Encryption
CIBC	www.cibconline.cibc.com	Canadian Imperial Bank of Commerce	RSA	RC4 128
RBC	www1.royalbank.com	Royal Bank of Canada	Verisign	RC4 128
TD	easyweb.tdcanadatrust.com	The Toronto Dominion Bank	RSA	AES-256
Scotiabank	www.scotiaonline.scotiabank.com	Bank of Nova Scotia	RSA	RC4 128
BMO	www1.bmo.com	Bank of Montreal	Entrust.net	RC4 128
PC Financial	www.txn.banking.pcfinancial.ca	Loblaw Companies LTD	RSA	RC4 128

Table 2.1: Comparing SSL certificate components

To understand CNs, a user must understand the domain name hierarchy, e.g., the important part of www.txn.banking.pcfinancial.ca is the domain pcfinancial.ca. Otherwise, CNs do not provide much useful information; attackers may generate SSL certificates with a CN such as www.pcfinancial.secure-banking.ca. The Organisation (O) items on some SSL certificates also differ from a bank's otherwise well-known name. For example, PC Financial's SSL certificate Organisation is "Loblaw Companies LTD." For average Internet users, encryption algorithm specifications and Certificate Authority (CA) names probably mean almost nothing. Users must also know that Verisign is a trustworthy CA, but Verisecuresign is probably an imaginary CA (generated by phishers). Also users must understand the browser security model, i.e., there are root certificates embedded with browsers from several (about 100) *trusted* CAs (trusted by browser providers – users have no role in such decisions); these embedded CA certificates are used to verify a given website's SSL certificate. Users must understand that not all CAs are trustworthy, at least not to the same level. Shortcomings of the browser security model are well known and have been exploited in the past; e.g., as described in an article after the Katrina disaster [77]. (Note that several shortcomings of SSL are in fact inherent to public key cryptography; see, e.g., Davis [65].)

Possible domain name conflicts. PC Financial's online banking URL is www.pcfinancial.ca. If a user types in www.pcfinancial.com, an under-construction website owned by directNIC is displayed. However, for CIBC customers, www.cibc.ca redirects to the regular CIBC site at www.cibc.com (the same is true for RBC, TD, BMO, and Scotiabank websites). The CIBC branch locator service is provided externally by a different company and the URL is cibc.via.infonow.net. Thus valid CIBC pages are sometimes served from domains other than cibc.com. The domains cibc.net and cibc.org are not owned by CIBC. The domain wwwcibc.com is also registered by someone other than the bank; this may trick many CIBC users if used in a phishing attack. Similar naming conflicts are quite common for other banks as well. If we take the possible similar domain names under other countries' top level domains, understanding domain names gets even more complicated. In fact, a domain name search at Netcraft's website (searchdns.netcraft.com) results in more than 500 entries with 'rbc' as part of a domain name. All banks suggest users look into the URL displayed on a browser's address bar to identify possible phishing sites. Nevertheless it is unclear what exactly users should look for, and how to interpret a domain name in the presence of such conflicting names. Assuming users understand complex domain name policies seems to be very far from a sound security principle.

2.3.2 Anti-malware Requirements

Most banks' customer agreements require users to install and maintain up-to-date copies of anti-virus, firewall, and anti-spyware programs. In this section, we discuss several issues related to banks' anti-malware requirements.

Cost of anti-malware. Banks advertise "free" anti-malware products on their websites although the free products are generally trial or detection-only versions. Banks recommend users to buy anti-malware from reputable vendors such as Symantec and

McAfee. Users must also pay for subscription renewal fees every year; in some cases subscription renewal is automatic and difficult to cancel [280]. The use of multiple computers to access online accounts multiplies the anti-malware cost. Thus evidently, users must spend a significant amount of money to be eligible to use *free* online banking services. For example, according to CIBC's anti-malware recommendation, a user must spend around $80 to buy discounted products from Symantec and Webroot. The monthly fee for maintaining a checking account at CIBC is approximately four dollars, thus yearly a user might be paying less than $50 for accessing all other banking services, e.g., bank tellers, bank machines, point-of-sale payments through a bank card, and telephone banking. Although there exist decent quality anti-virus, firewall and anti-spyware programs (e.g., `avast!`, `ZoneAlarm`, and `AdAware` respectively) which are offered for free for home users, banks do not mention those on their websites. These free anti-malware programs have been rated highly by many reviews, e.g., see `AV-test.org`.

In addition to anti-malware, banks also ask users to run various free security tools. For example, RBC recommends users to test computers using `Shields Up` (from Gibson Research) and Symantec `Security Check`. Shields Up is a web-based Internet vulnerability profiling tool to asses file sharing, common port vulnerabilities, messenger spam etc. Symantec Security Check is also a web-based tool for network vulnerability tests and virus detection; it requires installing an ActiveX control and Java run time environment. To fix any detected vulnerabilities or malware, users are advised to buy security products from McAfee/Symantec.

Usability and maintenance. Proper installation and maintenance of anti-malware requires time and a certain level of technical expertise. Maintenance tasks include downloading malware signature updates (although this is somewhat automated), product security updates (e.g., updates that fix known vulnerabilities within the product itself), and yearly license renewal. Sometimes users may have to deal with more complex issues such as failure of an auto-update or scheduled virus check. Completing such maintenance tasks correctly remains challenging for many average computer users. Banks do not take any responsibility for any difficulties, consequences or costs of installing and maintaining any recommended anti-malware. While reporting a possible fraud, CIBC users are asked to attach scan-logs from anti-virus and anti-spyware programs. How reliably an average Internet user can perform such operations is apparently an open question. Although banks advertise that online banking can be used from *anywhere*, users must ensure that up-to-date anti-malware programs exist in all computers used for online banking “including a computer at work, the library, an Internet cafe or another public place” (according to CIBC).

To clean a PC infected with known malware, users may need to follow one or more steps from the following list (note, this list is not comprehensive): (i) completely re-install the OS (as acknowledged by Microsoft [80]); (ii) check the infected PC’s hard disk by running anti-malware from another *malware-free* PC; and (iii) check the infected PC by running it in a *safe mode* (i.e., a reduced functionality mode). Performing any of these tasks is challenging for average users.

Despite the efforts of making personal firewalls user friendly, a cognitive study [122] of 13 most popular firewalls reveals several usability drawbacks of these products. (Effective use of enterprise firewalls is also subject to dispute, see e.g., Singer [244].) Like any other security products, a misconfigured personal firewall may endanger a user’s safety more than no firewall at all due to a false sense of security (cf. [35], [122]). One user study [158] of 378 U.S. respondents revealed a significant gap between the user-reported and actual state of security of their computers. For example, while 92% of participants reported to have up-to-date anti-virus (AV) software, in reality, only 51% had an AV signature file updated within the past week. Some statements regarding anti-malware programs, as provided on several banks’ websites, are also misleading and too broad in reality. For example, TD Canada states that firewalls allow only “the connections that are known and trusted,” which is generally far from the current Internet reality; e.g., firewalls may do nothing when a user visits a phishing site.

As a step towards the fight against spyware, RBC’s recommendations include: (i) Google search a product name to find out whether it contains any spyware; (ii) always carefully read licensing agreements and privacy policies of a product; and (iii) use anti-spyware if the user “suspects” any spyware activities, e.g., frequent pop-up advertisements, computer performance degradation. Only technically advanced users are likely to take advantage of Internet search to explore whether a file is malicious or contains embedded malware. Expecting users to read and understand software agreements seems unrealistic (see Section 2.3.3). Spyware may not be explicitly visible on a PC, and a computer's performance may degrade for several other reasons, e.g., disk fragmentation, diminishing free disk space, increased number of start-up programs. Therefore, it remains unclear how a user may "suspect" spyware infection on a given PC. Removing spyware from a computer may be "difficult" as stated by BMO; the bank encourages users to consult a trusted third party that specializes in computer maintenance and repair assistance (at their own expense).

In a user study by Downs et al. [70], 95% of users were familiar with the term "spyware;" 70% of the participants used online banking. Interestingly, several users believed that spyware was "protecting" their computer. Expecting such users to understand and follow a bank's anti-malware requirements may lead to unwanted consequences.

Shortcomings of anti-malware. Most current anti-virus (AV) programs are signature based, i.e., they mainly detect known malware. However, a signature is typically generated only when malware attempts to actively propagate in the wild. Small-scale attacks in which only a few computers are targeted, generally remain undetected by most anti-virus solutions. Commodity anti-virus vendors generally take an extended period of time to generate signatures of such malware, if at all. For example, Shipp [242] reported that a targeted trojan first released in June 2006 was detected by only four AV products in Oct. 2006; other AV products could not detect the malware even after months of its release. An AusCERT study [296] reported that popular AV products have an 80% miss rate on new malware as malcode authors generally test their malware against commonly used AV products before releasing it to the wild. In a malware trend report [56], Commtouch reported to detect $42,652$ distinct variants of the Storm-Worm in a period from Jan. 18-30, 2007, i.e., $3,824$ variants per day. Apparently it is very difficult for traditional anti-virus to generate so many signatures so quickly. In a recent (Aug. 2007) anti-virus test [63], only three out of ten AV products could catch all 18 (known) sample viruses. AV products often fail completely against malware that exploits zero-day vulnerabilities (i.e., *unknown* to the public domain). According to one study [7], as of June 16, 2007, the average lifetime of a zero-day bug is 348 days (with shortest and longest lifetime of 99 and 1080 days respectively). While AV products often fail to detect malware, sometimes legitimate programs are falsely identified as malicious, e.g., Symantec AV falsely detected malware on `Filezilla` and `NASA World Wind` [227]; such incidents further reduce users' trust on anti-malware.

Several recent instances of malware, when installed on a PC, e.g., by exploiting a zero-day vulnerability, attempt to remove or disable all common anti-malware on the infected PC. Some also change the `HOSTS` file on a Windows PC or otherwise poison a user's local DNS cache to redirect security-related websites, e.g., Windows update and McAfee, to marketing-related or malicious websites.

Virtual machine based rootkits, e.g., SubVirt [142], Blue Pill [219] may take complete control over a commodity OS, and may only be removed through a complete OS reinstallation. Rootkits installed on a graphics or network card [118] can even survive a low-level re-formatting of an infected hard disk following a full OS re-installation.

Attackers are also targeting security flaws in widely used anti-malware programs; one reason is that these programs generally run in a higher privilege mode than other applications. In a 15-month period ending Mar. 31, 2005, 77 separate vulnerabilities were discovered in security products from well-known vendors including Symantec, F-Secure and CheckPoint [295]. Symantec has disclosed 22 security advisories for its products in 2008.[6] In Nov. 2006, bot programs were reported to spread by exploiting known vulnerabilities in Windows and Symantec anti-virus (corporate edition) [236]. In one global snapshot[7] of current attacks as of Apr. 18, 2007, 70.5% of attacks were attributed to the Symantec anti-virus remote stack buffer overflow vulnerability (`CVE-2006-2630`) even though the vulnerability was reported on May 24, 2006, and a patch was promptly released. Evidently, users do not (or cannot) keep up with software updates (see Section 2.3.4), and their PCs may be compromised through their use of anti-malware tools. Of course, banks accept no responsibility of such unfortunate situations.

[6] For a list of Symantec product advisories, see `http://www.symantec.com/avcenter/security/SymantecAdvisories.html`.

[7] Arbor Network's ATLAS Dashboard `http://atlas.arbor.net`.

Banks also advise users to scan emails with an anti-virus program. Users are asked to be cautious about emails from unknown sources to prevent users from phishing attacks. However, phishing or spam email may be sent from a known contact's email address, and can even mimic patterns of a regular email from that contact [18]. Current anti-malware does not detect such attacks, and even most cautious users may fall prey to such an attack. We thus conclude that banks have unrealistic expectations with respect to users dealing with malware and spyware.

2.3.3 Documentation and Agreements

Banks strongly advise customers to review banking related user agreements including online banking, client card, and privacy agreements. Users' banking responsibilities are outlined in these agreements and on bank websites. Users must read and review these documents to understand important conditions and requirements of the 100% reimbursement guarantee. When we were setting up bank accounts with the six major banks, not one bank representative made us specifically aware of important online banking issues other than protecting the password (e.g., not to write down a password); we were assumed to have read and agreed to all terms and conditions as laid out on the agreements and websites. However, all banks provided us a printed copy of the agreements. Using the bank card or online banking confirms that "you have read and understood the agreement and agree to its terms and conditions" (as stated by RBC). Banks also state that agreements can be changed *at any time*, and users will be notified by "a notice on our website." We argue that many users do not read client agreements or security advice on bank websites, and therefore may remain unaware of the requirements they must fulfill to do online banking *safely.* In fact, the survey results in Section 2.4 support our conviction.

Banks also ask users to read software agreements carefully to avoid spyware/adware installation. However, RBC admits that information about spyware installation may be embedded in a third-party license agreement, and "These references may be hard to find and the user may not realize the full implications of the install." Grossklags and Good [111] conducted user studies evaluating the readability and usability of End User License Agreements (EULAs) using 50 popular software programs from download.com. The average EULA length was 2752 words. (In comparison, lengths of RBC's electronic access agreement,[8] bank card agreement, and privacy policy are about 5100, 3600, and 500 words respectively.) Only 1.4% of participants reported reading EULAs often and thoroughly, while 66.2% admitted to rarely reading or browsing the content of EULAs. It is also questionable how many users understood the implications of EULA content even when read thoroughly. In addition to software agreements, banks also advise users to check privacy policies of websites where a user may provide sensitive personal information (e.g., PayPal). RBC also asks users to check third-party security bulletins regularly, especially for OS and browser security updates. We conclude that banks have unrealistic expectations with respect to users reading and understanding legal and technical documents including online banking agreements, account policies, and EULAs.

2.3.4 Software Updates

Banks strongly advise users to keep an OS and browser up-to-date with security patches. Nonetheless, out-dated browser versions such as IE 5.5 and Firefox 1.0 remain listed as supported by most banks. Beyond the web browser, generally there are many more applications commonly used by millions of users, and thereby being targeted by attackers. Microsoft Office products such as Word and Excel are popular targets. Media players such as Windows Media Player, Realplayer, iTunes, and Winamp also pose security threats if unpatched. Vulnerabilities in Instant Messaging (IM) and desktop email clients have also been widely exploited. In fact, users must keep *all* applications up-to-date to avoid known attacks – a daunting task even for advanced users, especially in a Windows environment. As reported from a survey [235] of 20,000 Windows PCs, less than 2% of those were found to be fully patched. Other operating systems, e.g., Ubuntu Linux and Mac OS X provide an update mechanism to keep all installed (native) software packages updated – but these are used by less than 5% of the population.

After installing a new OS, updating the OS and other software on the computer is not the first task that a user might want to do. However, an Internet connected PC

[8] From https://www.rbcroyalbank.com/onlinebanking/bankingusertips/agreement/termsindex.html (Apr. 3, 2007).

may survive only minutes[9] before being compromised via unpatched vulnerabilities. A Windows XP installation requires downloading 70 to 260 MB of security updates from Microsoft [164] (a 13-page SANS report [224] provides guidelines for surviving the first day of an XP installation). Downloading such large updates is highly problematic for users with a dial-up or slow-speed Internet connection.

Updating OS, browser, firewall and anti-malware is challenging for many Internet users. *Patch management* includes collecting all necessary patches, dealing with post-patch conflicts, and determining the trustworthiness of a patch source [35], which is a difficult problem even for enterprise IT departments. In addition to usability problems, such updates may even frustrate or fool diligent users. For example, Bellissimo et al. [30] showed that some popular software updates (e.g., McAfee VirusScan) were vulnerable to man-in-the-middle attacks; i.e., a malicious party could install malware exploiting several software update vulnerabilities. Malware such as `Trojan-Dropper.MSWord.Lafool.v` [140] even attempts to propagate as a security update alert from reputable vendors (e.g., McAfee). Some critical security patches released by Microsoft may even crash a system or make it unusable [79]. The critical Windows XP SP2 update stopped many programs including anti-virus from working properly [267]. A Symantec signature update in May 17, 2007 falsely identified (and deleted) two critical system files of the Chinese edition of Windows XP SP2 as trojans, and thereby failed thousands of PCs to boot [59]; even though Symantec fixed the bug quickly, many users had to go through XP's recovery console (a not-so-user-friendly command-line tool) to fix the OS. Malware that upgrades exploiting Windows Update have also been reported in the wild [25].

Only 7 days in 2004 were without an unpatched known vulnerability in IE – the browser recommended by all banks; i.e., IE was unsafe 98% of the days in 2004 [231]. Even if IE or other popular browsers have improved their security more recently – which is arguable – this seems untrue of application software in general on users' PCs. Overall, while the research community recognizes patching as an open problem,[10] banks assume average home users can adequately deal with it.

[9] The average time between attacks is reported as 10 minutes (Nov. 15, 2008) at `http://isc.sans.org/survivaltime.html`.

[10] However, patching is as old a problem as software, e.g., see Glass [98].

2.3.5 User Authentication and Other Issues

We now briefly discuss online user authentication and some other important security and privacy related issues.

User authentication. Canadian banks largely rely on user-chosen passwords and Personal Verification Questions (PVQs) for online authentication. We limit our discussion here as the usability of passwords has been well-studied by others (e.g., Sasse and Adams [4]). Table 2.2 compares online password and PVQ requirements of different banks. Passwords are case-sensitive, but some banks restrict use of special characters (e.g., #, @). BMO allows only numeric passwords of length six. Fixed-length and small upper limits (e.g., eight) on password length create usability problems.

	Password	PVQ answer
CIBC	6-12	4-21 (2 PVQs)
RBC	8-32	4-20 (3 PVQs)
TD	5-8	functionality absent
Scotiabank	8-16	functionality absent
BMO	6	functionality absent
PC Financial	6-12	1-20 (3 PVQs)

Table 2.2: Comparing password and PVQ answer length across six banks

Banks recommend that all passwords and PVQs be unique. Most people use several password-protected accounts,[11] and many reported having multiple bank accounts in our survey (Section 2.4). It is generally difficult to create and memorize many unique secrets [4]. Banks strongly recommend (but do not force) users to change passwords as frequently as every month (e.g., PC Financial). Some banks force users not to reuse a recently used password. Sasse et al. [230] reported that login failures increase after a password change as the new password interferes with the old one. Most users are also reported to use a word with a number at the end when a frequent password change is enforced [229]. Such a password strategy may help attackers to design more efficient password crackers [292]. Banks' policies require that users logout from online banking when a banking session is over. However, users who navigate

[11] A large scale (over half a million participants) survey [90] in 2007 reported that an average user has about 25 password-protected accounts with 6.5 unique passwords as reported (see also [96]).

away from online banking through links on online banking pages may not see the sign-out button.

Besides strong recommendations such as password uniqueness, CIBC asks users to promptly reset a banking password over telephone or from a *trusted* computer, after accessing online banking from a public computer. Despite such advice, most banks allow obviously *weak* passwords, e.g., '123456' and '111111.' An example of a "rock-solid" password according to RBC[12] is `iwthyh` (mnemonic of The Beatles' "I want to hold your hand"); but this does not even meet RBC's password length requirement. Although recommended [4] (see also [292]), no bank provides feedback on the strength of a user selected password (cf. [162]).

Users can set up PVQs for resetting a forgotten password (e.g., CIBC, RBC, PC Financial) or as part of the login process (e.g., RBC, CIBC). Online password reset reduces help desk calls (and therefore costs) for banks, and is also convenient for users at the usability cost of memorizing more secrets (cf. Section 6.3.2). As PVQs are rarely used, i.e., users do not need to recall the *secret* answers often, users tend to choose easily memorable answers. A PVQ answer is typically case-insensitive, free of any special characters, and shorter in length than a password. A 6-character password-protected CIBC account can be accessed by answering two PVQs, as short as 4 characters each. Interestingly, a 6-character case-sensitive password can be overridden by a 3-character case-insensitive secret as implemented by PC Financial; there are three PVQs, but each PVQ answer can be as short as a single character. In effect, PVQ answers are allowed to be weaker than passwords, although PVQs are as good as a password to get into an account. Answers to PVQs (e.g., mother's maiden name) may easily be guessed by close contacts of a user as reported even in a pre-Internet era user study by Zviran and Haga [300] (see also [136]). The abundance of personal information on the web (or otherwise available online) enables even complete strangers to make informed guesses (e.g., [152]). Authentication guidelines of the Office of the Privacy Commissioner of Canada [191], recommend that authentication secrets should not be derived from personal identity facts.

[12] Accessed Apr. 24, 2007, from `http://www.rbc.com/security/safe_passwords.html`.

Anti-phishing email tips. To distinguish phishing emails from real CIBC emails, users are asked to look for personalized email messages (e.g., customer's real name on the message). Attackers may collect public domain information for deploying targeted phishing attacks (also known as *spear phishing*). User accounts in social networking websites such as MySpace have been compromised in the past to extract personal information [183]; such information could be used to launch spear phishing attacks. In reality, there have been reported cases of fake Microsoft Update patch email with malicious URLs, including the targeted users' full names in the email body [225]. CIBC reports that 'Properties' of the sender address in an email shows the 'actual' email address, although everything in an email header can be spoofed. RBC suggests users look for misspelled words, and distorted images in emails. RBC also warns users to be suspicious about websites that collect confidential information but are not SSL-enabled. These recommendations are of little help as phishing attacks are now more professional; for example, some scam-spammers are reported [37] to make use of sophisticated *mind game* tricks, and some phishing sites are even SSL-protected [181]. In fact, a Microsoft sponsored survey [15] of $2,482$ American adults revealed that 58% of the participants were not at all aware of online threats, while 17% had fallen victim to some forms of online frauds.

Scotiabank sends news and helpful tips to users through 'The Vault' mailing list. The bank does not illustrate how users can verify the authenticity of emails received through the Vault. These emails contain live (click-able) links and users are encouraged to subscribe to the Vault; there is a chance to win $1000 every month for subscribers. Such practices may help phishers in several ways, e.g., phishing email may be sent to users stating that they have won the Vault prize, and need to sign-on to Scotiabank online (actually a spoofed website) to collect the prize.

Several banks (e.g., CIBC, RBC) provide a *secure message centre* for sending important messages/notifications to users. Users can also send messages, e.g., banking instructions or questions, to banks using the message centre. The message centre can only be accessed after logging into a bank website. This is apparently a more secure way of communication than regular email. However, banks may notify users through regular email when a new message is posted on the message centre. As viewing messages from the message centre requires login to online banking, such notifications via email may open a new avenue to phishing attacks.

Unnecessary information collection. Banks suggest that users not provide their Social Insurance Number (SIN) as an identification token when other types of identifiers may be sufficient. However, we were asked by most banks to provide the SIN when opening even a regular checking account. Although we declined to provide the SIN, all accounts were opened successfully. Bank representatives also asked for several other unnecessary personal information, e.g., family size, income, rent, dependents, which were in no way related to a checking account. When asked, bank representatives told us they were collecting such information which might be required to "better serve" us in the future, e.g., if we apply for credit cards. However, we argue that such unnecessary information collection should be avoided as it may pose increasing risks to privacy and security, and allows banks to collect information (for marketing purposes) which users may misunderstand as being required by banking standards to open new accounts.

2.4 User Survey

We conducted a survey to gain insight on user compliance with online banking requirements. In this section we discuss the results. We compiled a questionnaire (attached in Appendix A.1) from some common requirements and recommendations of Canadian online banking. This was reviewed and approved by our university ethics committee. The survey was anonymous and voluntary.

Participants and results. There were 123 participants in the survey. Our participants include under-graduate (3rd and 4th year) and graduate students, professors, post-doctoral fellows, network administrators, security researchers and professionals, mostly from the computer science department of our university. Participants in this study are not representative of general online banking users; but we conducted the survey to understand whether highly technical and security-aware users fulfill a subset of banks' requirements and recommendations. In Appendix A.2, we provide our survey data and discuss the findings in detail. Note that other than their honesty, participants were in no way motivated to complete the survey or answer correctly. We presented a preliminary version of this work in a graduate course, and some participants mentioned that they attended the class talk and their online banking habits were thereby influenced (e.g., they became more aware of the banking requirements – thus if anything, our results over-report actual security compliance). For some questions a few users chose the “Don’t know” option. We did not include those answers to the results. We allowed space for comments on the questionnaire. Some other requirements and recommendations which have not been discussed earlier in this chapter were also included in the survey; these include file sharing through Windows or peer-to-peer (P2P) clients, clearing browser cache and closing the browser after a banking session, and checking bank statements.

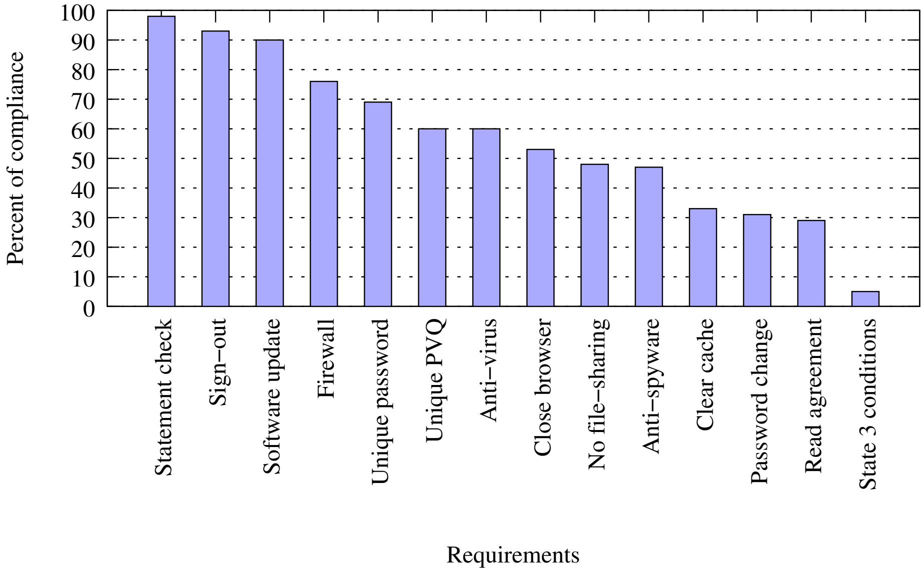


Figure 2.1: Summary of conformance

Summary of results. Figure 2.1 summarizes our findings. Many users reported using Firefox/Mozilla; we did not collect data on versions. Most IE users use IE6 which is less secure than IE7 (according to Microsoft). Many users use Linux, but other than RBC, no banks explicitly mention supporting it. A significant portion (about 50%) of participants do not comply with anti-virus, close-browser, and unique password/PVQ requirements (or recommendations). The majority (more than 50%) do not comply with the requirements (or recommendations) regarding anti-spyware, file-sharing, clear-cache, password change, and reading agreement. Very few participants were able to state three conditions for the 100% reimbursement guarantee. Almost all participants sign-out from online banking, and check their bank statements. Firewalls are deployed by a good portion of the participants. Most also reported to keep their OS, browser, and anti-malware software up-to-date; note, however, that many IE6 users did not upgrade to IE7.

2.5 Concluding Remarks

Banks advertise that users can start online banking “in minutes.” However, to comply with the security requirements and recommendations, we expect most users would be delayed hours or days, if indeed technically capable of doing so at all. Banks state that security is a “shared responsibility.” Our analysis and survey suggest that the users’ share of this responsibility is large and unrealistic given the current Internet environment and available technologies. Most participants did not fulfill all the listed requirements in our survey. We argue that if such predominantly technical and security-aware participants fail to satisfy online banking requirements, expecting average home users to meet all such requirements is extremely naïve. Therefore, we conclude that most average users are ineligible for the 100% reimbursement guarantee banks assert, and doing online banking with “confidence” and “peace of mind” is no more than a marketing slogan which misleads users.

Indeed, as simply one recent example, in January 2007, CIBC reported the loss of a computer hard drive with unencrypted personal financial information (including name, address, social insurance number, date of birth, signature) of about $470,000$ mutual fund customers [178].[13] Many Canadian customers are also affected by the recently-reported (Mar. 2007) TJX data breach [58] of about 45 million users. In customer agreements, banks do not indicate how they would notify or compensate users in such incidents. Note that Canadian banks are currently not legally bound to disclose such breaches. Banks also do not specify whether users will be reimbursed other than for monetary losses, e.g., time lost in recovering from financial fraud and

[13] See `http://www.caslon.com.au/datalossnote2.htm` for well-known consumer data losses from major banks.

theft. Banks discourage users using P2P file-sharing from the same PC as they perform online banking with; ironically, analyzing P2P traffic for inadvertently disclosed sensitive files of top 30 US banks, Johnson et al. [135] reported to discover a significant number of files with confidential banking information, including a spreadsheet with personal details of $23,000$ business accounts, and a detailed manual of a bank's security review process.

As stated in an updated personal account agreement (effective from April 1, 2007), CIBC may close a customer's bank account "without notice...to prevent future losses if you are a victim of fraud." Thus a defrauded customer may lose her bank account as a consequence of using the heavily advertised *free* online banking service, and thereby becoming a *double victim.* Banks emphasize that as long as users maintain the "security" of a bank card number and password, no one can gain access to their online banking accounts. One bank (CIBC) also assures customers that the bank will not "provide [any] service that compromises the security and confidentiality of customer information." In contrast, new malware attacks (*bank-stealing Trojans* or *session-hijacking*, e.g., Win32.Grams [46]; see also CERT [165]) can perform fraudulent transactions in real-time after a user has logged into an online banking account. Banks do not specifically advise users how to protect against such attacks, nor do they explain how unique "rock-solid" passwords, 128-bit SSL encryption or other "enhanced" security techniques may help users against these attacks. In fact, most existing or proposed solution techniques are susceptible to these new attacks (e.g., including [195] and two-factor authentication such as a password plus a passcode generator token). Banks may reimburse any money lost due to online banking, at least when users meet banks' requirements; however, users pay with their own personal time and mental energy to address consequences of credit-card fraud and identity theft as enabled in part by the use of online banking. In general, we argue that several security-sensitive online services are now being offered (and even heavily pushed) to average home users without the availability of appropriate *usable* tools to perform those tasks safely. The growing disconnect between service providers' expectations, and technical capabilities of the general user population, may result in increasing loss of trust in the web over the long run.

Chapter 3

Authentication from an Untrusted Computer

In this chapter, we propose a protocol called *MP-Auth* (Mobile Password Authentication) for authentication and integrity protection for sensitive online transactions from a potentially untrusted computer. MP-Auth also aims to address semantic attacks such as phishing. MP-Auth is an attempt to make everyday users free from following a comprehensive list of security *best practices* (as discussed in Chapter 2) for performing sensitive online tasks, e.g., Internet banking.

3.1 Introduction

In the current Internet environment, most consumer computers are infected with one or more forms of spyware or malware. Internet connected PCs are not *safe* anywhere; an improperly patched home or public computer generally survives only minutes.[1] Semantic attacks such as phishing also widely target general Internet users. Software keyloggers are typically installed on a user PC along with common malware and spyware [171]. An increasing number of phishing sites also install keyloggers on user PCs, even when users do not explicitly download or click any link on those sites [204]. Many of these attacks attempt to extract user identity and sensitive account information for unauthorized access to users' financial accounts; for example, user names and passwords for thousands of bank accounts have been found on an online storage site reportedly gathered by a botnet [210]. Passwords enjoy ubiquitous use for online authentication even in such an environment, although many more secure (typically also more complex and costly) authentication protocols have been proposed. Due to the usability and ease of deployment, most North American financial transactions over the Internet are still authenticated through only a password. Hence passwords

[1] An average time between attacks of 5 minutes, as of Aug. 2, 2008, is reported at `http://isc.sans.org/survivaltime.html`.

are a prime target of attackers, for economically-motivated exploits including those targeting online bank accounts and identity theft.

As one example of highly sensitive Internet services, online banking often requires only a bank card number (as *userid*) and password for authentication. Users input these credentials to a bank website to access their accounts. An attacker can easily collect these long-term secrets by installing a keylogger program on a client PC, or embedding a JavaScript keylogger [214] on a compromised website. As plaintext sensitive information is input to a client PC, malware on the PC has instant access to these (reusable) long-term secrets. We argue (as do others – e.g., see Laurie and Singer [148], Kursawe and Katzenbeisser [146]) that for some common applications, passwords are too important to input directly to a typical user PC on today's Internet; and that the user PC should no longer be trusted with such plaintext long-term secrets, which are intended to be used for user authentication to a remote server. Additionally, phishing attacks can collect plaintext reusable userid-password pairs even if a user's PC is malware-free (through e.g., domain name hijacking [124], or the Kaminsky DNS-flaw [137]).

To safeguard a long-term password, we build on the following simple idea: use a hand-held personal device, e.g., a cellphone or PDA to encrypt the password (combined with a server generated random challenge) under the public key of an intended server, and relay through a (possibly untrusted) PC only the encrypted result in order to login to the server website. This simple challenge-response effectively turns a user's long-term password into a one-time password in such a way that long-term passwords are not revealed to phishing websites, or keyloggers on the untrusted PC.

The resulting protocol, called *MP-Auth* (short for *M*obile *P*assword *Auth*entication), is proposed primarily to protect a user's long-term password input through an untrusted (or untrustworthy) client PC. The use of a mobile device in MP-Auth is intended to protect user passwords from easily being recorded and forwarded to malicious parties, e.g., by keyloggers installed on untrustworthy commodity PCs. For usability and other reasons, the client PC is used for the resulting interaction with the website, and performs most computations (e.g., session encryption, HTML rendering etc.) but has access only to temporary secrets. The capabilities we require from a mobile device include encryption, alpha-numeric keypad, short-range network connection (wire-line or Bluetooth), and a small display. Although we highlight the use of a cellphone, the protocol can be implemented using any similar *trustworthy* device (e.g., PDAs or smart phones), i.e., one free of malware. There are known attacks against mobile devices [101], but the trustworthiness of such devices is currently more easily maintained than a PC, in part because they contain far less software; see Section 3.3.3 for further discussion on mobile device security. Note that, despite our use of a personal device in conjunction with a PC, MP-Auth is essentially a single-factor (password-only) authentication protocol; we do not store any other secrets on the device.

To protect a password from being revealed to a phishing site, the password is encrypted with the *correct* public key of an intended website (e.g., a bank). Thus MP-Auth protects passwords from keyloggers and various forms of phishing attacks (including deceptive malware, DNS-based attacks or pharming, as well as false bookmarks). MP-Auth also protects against session hijacking, by providing transaction integrity through a transaction confirmation step. Unlike standard two-factor techniques, MP-Auth does not store any secret on the mobile device.

Phishing attacks have been discussed in the literature since 1997 (see [87]); however, few, if any, anti-phishing solutions exist today that are effective in practice. In addition to several anti-phishing proposals (e.g., [214], [293]), there exist many software tools for detecting spoofed websites (e.g., eBay toolbar, SpoofGuard, Spoofstick, Netcraft toolbar). However, most of these are susceptible to keylogging attacks,[2] and phishing toolbars are barely effective in reality [297]. On the other hand, several authentication schemes which use a trusted personal device, generally prevent keyloggers, but do not help against phishing or session hijacking attacks. New malware attacks (*bank-stealing Trojans* or *session-hijacking*, e.g., Win32.Grams [46], Trojan.Silentbanker [255]; see also CERT [165], web-rootkits [130]) attempt to perform fraudulent transactions in real-time after a user has logged in, instead of collecting userids and passwords for later use. Most existing or proposed solution techniques are susceptible to these new attacks, including e.g., Phoolproof [195] and two-factor

[2] PwdHash [214] can protect passwords from JavaScript keyloggers, but not software keyloggers on client PCs.

authentication such as a password and a passcode generator token (e.g., SecurID). In contrast, the primary goals of MP-Auth are twofold: protect passwords from both keyloggers and phishing sites, and provide transaction integrity.

Our contributions. We propose MP-Auth, a protocol for online authentication using a personal device such as a cellphone in conjunction with a PC. The protocol provides the following benefits without requiring a trusted proxy (e.g., [53]), or storing a long-term secret on a cellphone (e.g., Phoolproof [195]).

1. Keylogging Protection. A client PC does not have access to long-term user secrets. Consequently keyloggers (software or hardware) on the PC cannot access critical passwords.
2. Phishing Protection. Even if a user is directed to a spoofed website, the website will be unable to decrypt a user password. Highly targeted phishing attacks (*spear phishing*) are also ineffective against MP-Auth.
3. Pharming Protection. In the event of domain name hijacking [124, 137], MP-Auth does not reveal a user's long-term password to attackers. It also protects passwords when the DNS cache of a client PC is poisoned.
4. Transaction Integrity. With the transaction confirmation step (see Section 3.2) in MP-Auth, a user can detect any unauthorized transaction during a login session, even when an attacker has complete control over the user PC (through e.g., SubVirt [142] or Blue Pill [219]).
5. Applicability to ATMs. MP-Auth is suitable for use in ATMs (automated teller machines), if an interface is provided to connect a cellphone, e.g., a wireline or Bluetooth interface. This can be a step towards ending several types of ATM fraud (see Bond [38] for a list of ATM fraud cases).

A comprehensive survey of related authentication schemes used in practice and/or proposed to date, and a comparative analysis of these to MP-Auth are provided in Chapter 4. We analyzed MP-Auth using the Automated Validation of Internet Security Protocols and Applications (AVISPA [14]) protocol analysis tool; no attacks were found. A formal proof sketch of MP-Auth using the Protocol Composition Logic (PCL) [64, 117, 216] is also provided. We have also implemented a prototype of MP-Auth for performance testing.

Organization. The MP-Auth protocol, threat model and operational assumptions are discussed in Section 3.2. A brief informal analysis of MP-Auth messages, discussion on how MP-Auth prevents common attacks, and circumstances under which MP-Auth fails to provide protection are outlined in Section 3.3. Discussion on usability and deployment issues related to MP-Auth are provided in Section 3.4. The performance and basic implementability of MP-Auth is discussed in Section 3.5. Section 3.6 concludes. Appendix B.1 provides AVISPA test code for MP-Auth and related discussion. The PCL analysis of MP-Auth is provided in Appendix B.2.

3.2 Mobile Password Authentication (MP-Auth)

In this section, we describe the MP-Auth protocol, including threat model and operational assumptions, password renewal, and public key installation methods.

Threat model and operational assumptions. The primary goals of MP-Auth are to protect user passwords from malware and phishing websites, and to provide transaction integrity. We assume that a bank's *correct* public key is available to users (see below for discussion on public key installation), and mobile devices are malware-free. Public keys of each target website must be installed on the device. (We assume that there are only a few financially critical websites that a typical user deals with.) A browser on a PC uses a bank's SSL certificate to establish an SSL connection with the bank website (as per common current practice). The browser may be duped to go to a spoofed website, or have a wrong SSL certificate of the bank or the verifying certificating authority. The protocol does not protect user privacy (of other than the user's password) from an untrusted PC; the PC can record all transactions, generate custom user profiles etc. Visual information displayed to a user on a PC screen is also not authenticated, i.e., a malicious PC can display misleading information to a user without being (instantly) detected. Denial-of-service (DoS) attacks are not addressed. A communication channel between a personal device and PC is needed, in such a way that malware on the PC cannot infect the personal device.[3]

[3] The first *crossover* virus was reported [166] in February 2006.

U, M, B, S	User, a cellphone, a browser on the untrusted user PC, and the server, respectively.
ID_S, ID_U	Server ID and user ID, respectively. ID_U is unique in the server domain.
P	Long-term (pre-established) password shared between U and S.
R_S	Random number generated by S.
$\{data\}_K$	Symmetric (secret-key) authenticated encryption (see e.g., [99], [28]) of $data$ using key K.
$\{data\}_{E_S}$	Asymmetric (public-key) encryption of $data$ using S's long-term public key E_S.
$X.Y$	Concatenation of X and Y.
K_{BS}	Symmetric encryption key shared between B, S (e.g., an SSL key).
$f(\cdot)$	A cryptographically secure hash function.

Table 3.1: Notation used in MP-Auth

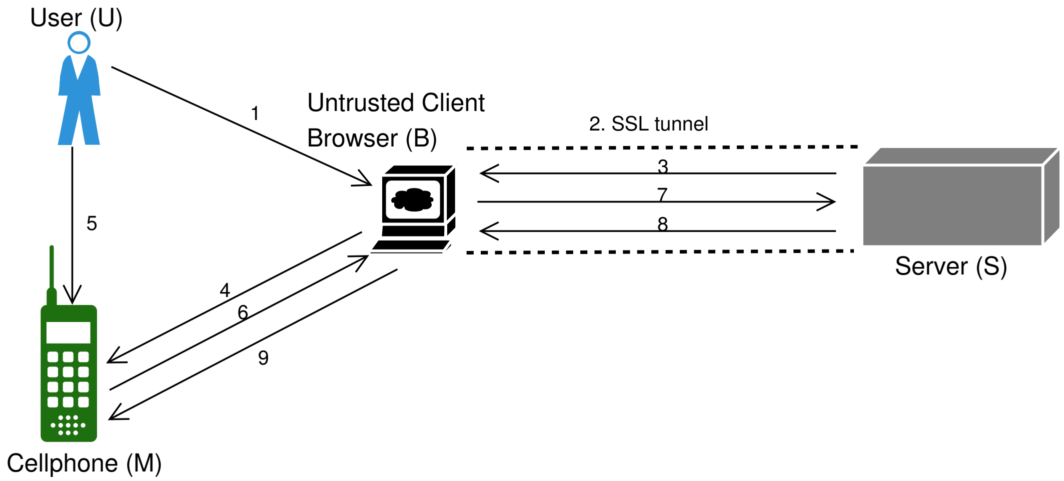


Figure 3.1: MP-Auth protocol steps

Protocol steps in MP-Auth. For notation see Table 3.1. Before the protocol begins, we assume that user U's cellphone M is connected to B (via wire-line or Bluetooth). The protocol steps are described below (see also Fig. 3.1).

1. U launches a browser B on the untrusted PC, and visits the bank website S.

2. B and S establish an SSL session; let K_{BS} be the established SSL secret key.

3. S generates a random nonce R_S, and sends the following message to B.

$$B \leftarrow S: \ \{ID_S.R_S\}_{K_{BS}} \tag{3.1}$$

4. B decrypts message (3.1) and forwards it to M.

$$M \leftarrow B: \; ID_S.R_S \tag{3.2}$$

5. M displays ID_S, and prompts the user to input the userid and password for S. A userid (e.g., bank card number) may be stored on the cellphone for convenience; the password should not be stored or auto-remembered.

6. M generates a random secret nonce R_M and encrypts R_M using E_S. M calculates the session key K_{MS} and sends message (3.4) to B (here, the userid ID_U is, e.g., a bank card number).

$$K_{MS} = f(R_S.R_M) \tag{3.3}$$

$$M \rightarrow B: \; \{R_M\}_{E_S}.\{f(R_S).ID_U.P\}_{K_{MS}} \tag{3.4}$$

7. B (via SSL) encrypts message (3.4) with K_{BS}, and forwards the result to S.

8. From message (3.4), after SSL decryption, S decrypts R_M using its corresponding private key, calculates the session key K_{MS} (as in equation (3.3)), decrypts the rest of message (3.4), and verifies P, ID_U and R_S. Upon successful verification, S grants access to B on behalf of U. S sends the following message for M to B (indicating login success).

$$B \leftarrow S: \; \{\{f(R_M)\}_{K_{MS}}\}_{K_{BS}} \tag{3.5}$$

9. B forwards $\{f(R_M)\}_{K_{MS}}$ to M. M decrypts to recover $f(R_M)$ and verifies its local copy of R_M. Then M displays success or failure to U.

Transaction integrity confirmation. In MP-Auth, M and S establish a session key K_{MS} known only to them; malware on a user PC has no access to K_{MS}. Attackers may modify or insert transactions through the untrusted PC. To detect and prevent such transactions, MP-Auth requires explicit transaction confirmation by U (through M). The following messages are exchanged (after step 9) for confirmation of a transaction with summary details T (R_{S1} is a server generated random nonce, used to prevent replay).

$$M \xleftarrow{\{T.R_{S1}\}_{K_{MS}}} B \xleftarrow{\{\{T.R_{S1}\}_{K_{MS}}\}_{K_{BS}}} S \tag{3.6}$$

$$M \xrightarrow{\{f(T.R_{S1})\}_{K_{MS}}} B \xrightarrow{\{\{f(T.R_{S1})\}_{K_{MS}}\}_{K_{BS}}} S \tag{3.7}$$

M displays T to U in a human-readable way (e.g., "Pay \$10 to Vendor V from checking account C"), and asks for confirmation (yes/no). When the user confirms T, the confirmation message (3.7) is sent from M to S (via B). From message (3.7), S retrieves $f(T, R_{S1})$, and verifies with its local copy of T and R_{S1}. Upon successful verification, T is committed. Instead of initiating a confirmation step after each transaction, transactions may be confirmed in batches (e.g., four transactions at a time); then, T will represent a batch of transactions in the above message flows. The user-interface (UI) design of the confirmation step is extremely important to reduce dangerous errors; see, for example, Uzun et al. [269] for a comparative study and user-testing of different approaches to such UI design in the context of secure device pairing.

Some transactions may not require confirmation from the mobile device. For example, adding a new user account or setting up an online bill payment for a phone company should require user confirmation, but when paying a monthly bill to that account, omitting the confirmation step would seem to involve little risk. Similarly, fund transfers between user accounts without transaction confirmation may be an acceptable risk. A bank may configure the set of sensitive transactions that will always require the confirmation step. Deciding to omit the requirement of explicit confirmation should be done with hesitation. As reported in a Washington Post article [274], attackers compromised customers' trading accounts at several large U.S. online brokers, and used the customers' funds to buy thinly traded shares. The goal is to boost the price of a stock they already have bought and then to sell those shares at the higher price. This incident indicates that seemingly innocuous transactions may be exploited by attackers if transactions requiring confirmation are not diligently selected by banks.

Password setup/renewal. In order to secure passwords from keyloggers during password renewal, we require that the password is entered through the cellphone keypad. We assume that the initial password is set up via a trustworthy out-of-band method (e.g., regular phone, postal mail), and U attempts a password renewal after successfully logging into S (i.e., K_{MS} has been established between M and S). The following message is forwarded from M to S (via B) during password renewal (P_{old} and P_{new} are the old and new passwords respectively).

$$M \xrightarrow{X,\text{ where } X = \{ID_U.P_{old}.P_{new}\}_{K_{MS}}} B \xrightarrow{\{X\}_{K_{BS}}} S \tag{3.8}$$

Public key installation. One of the greatest practical challenges of deploying public key systems is the distribution and maintenance (e.g., revocation, update) of public keys. MP-Auth requires a service provider's public key to be distributed (and updated as needed) and installed into users' cellphones. The distribution process may vary depending on service providers; we recommend that it not be primarily Internet-based. Considering banking as an example, we visualize the following key installation methods (but emphasize that we have not user-tested these for usability):

1. at a bank branch, during an account setup (see Section 3.4 for usability issues).
2. through in-branch ATM interfaces (hopefully free of fake ATMs).
3. through a cellphone service (authenticated download) as data file transfer.
4. (for web-only banks) through removable flash memory card for cellphones (e.g., microSD card) mailed to users, containing the public key. In this case, one might consider the cost of an attack involving fake mailings to users.

A challenge-response protocol or integrity cross-checks (using a different channel, e.g., see [270]) should ideally be used to verify the public key installed on a cellphone, in addition to the above procedures. For example, the bank may publish its public key on the bank website, and users can cross-check the received public key, e.g., comparing *visual hashes* [199] or *public passwords* [114]; to reduce usability issues, an automated cross-check would be preferable.

Authentication without a personal device. It may happen that while traveling, a user may lose her cellphone, and cannot use MP-Auth to log in to her bank. As a secondary authentication technique in such emergency situations, one-time codes could be used. For example, banks may provide users a list of pass-codes (e.g., 10 digit numbers) printed on a paper which can be used for secondary login; i.e., a userid and pass-code entered directly on a bank login page allows emergency access to a user's account. However, such logins should be restricted to a limited set of transactions, excluding any sensitive operations such as adding a payee or changing postal address (cf. *TwoKind* authentication [19]).

3.3 Security and Attack Analysis

In this section, we first consider an informal security analysis of MP-Auth. We motivate a number of design choices in MP-Auth messages and their security implications, and discuss several attacks that MP-Auth is resistant to. We also list successful but less likely attacks against MP-Auth.

As a confidence building step, we have tested MP-Auth using the AVISPA [14] protocol analysis tool, and found no attacks. AVISPA is positioned as an industrial-strength technology for the analysis of large-scale Internet security-sensitive protocols and applications. AVISPA test code for MP-Auth and discussion are provided in Appendix B.1. The test code is also available online [153]. A formal proof sketch of MP-Auth using the Protocol Composition Logic (PCL) [64, 117, 216] is also provided in Appendix B.2.

3.3.1 Partial Message Analysis and Motivation

Here we provide motivation for various protocol messages and message parts. In message (3.1), S sends a fresh R_S to B, and B forwards ID_S, R_S to M. ID_S is included in message (3.2) so that M can choose the corresponding public key E_S. When U starts a session with S, a nearby attacker may start a parallel session from a different PC, and grab M's response message (3.4) (off-the-air, from the Bluetooth connection) to login as U. However, as S generates a new R_S for each login session (i.e., U and the attacker receive different R_S from S), sending message (3.4) to S by any entity other than B would cause a login failure.

The session key K_{MS} shared between M and S, is known only to them. Both M and S influence the value of K_{MS} (see equation (3.3)), and thus a sufficiently random K_{MS} is expected if either of the parties is honest (as well as capable of generating secure random numbers); i.e., if a malicious party modifies R_S to be 0 (or other values), K_{MS} will still be essentially a random key when M chooses R_M randomly. To retrieve P from message (3.4), an attacker apparently must guess K_{MS} (i.e., R_M) or S's private key. If both these quantities are sufficiently large (e.g., 160-bit R_M and 1024-bit RSA key E_S) and random, an offline dictionary attack on P becomes computationally infeasible. We encrypt only a small random quantity (e.g., 160-bit) by E_S, which should always fit into one block of a public key cryptosystem (including elliptic curve). Thus MP-Auth requires only one public key encryption. Browser B does not have access to K_{MS} although B helps M and S establish this key. With the transaction integrity confirmation step, all (important) transactions must be confirmed from M using K_{MS}; therefore, any unauthorized (or modified) transaction by attackers will fail as attackers do not have access to K_{MS}.

Analysis of simplified authentication messages. For the authentication phase in MP-Auth, the browser simply forwards messages between the personal device and the web server. Therefore for analysis, we simplify the MP-Auth authentication messages (steps 1 through 9 in MP-Auth, see Section 3.2) in the following way.

$$M \leftarrow S : ID_S.R_S \tag{3.9}$$

$$M \rightarrow S : \{R_M\}_{E_S}.\{f(R_S).ID_U.P\}_{K_{MS}}, \text{ where } K_{MS} \;=\; f(R_S.R_M) \tag{3.10}$$

$$M \leftarrow S : \{f(R_M)\}_{K_{MS}} \tag{3.11}$$

We assume that before the protocol run, M and S establish a shared secret P, and M has an authentic copy of S's public key E_S (e.g., obtained using an out-of-band method). At the end of the protocol run, the goals are to achieve mutual authentication between M and S, and establish a fresh session key known only to M and S. We assume that all protocol messages can be intercepted, modified, and stored by an attacker. Also the attacker can attempt to impersonate either M or S. However, none other than M and S knows P, and the private key corresponding to E_S is known only to S.

At the end of message (3.9), S believes that it has sent a fresh nonce R_S to M. M receives R_S, but it has no assurance of the true identity of S. For message (3.10), M generates a fresh nonce R_M, and an encryption key K_{MS} from R_M and R_S. M believes that K_{MS} is a fresh encryption key as R_M is freshly generated by M. M encrypts R_M using E_S so that none other than S can generate K_{MS} and learn P. When S verifies P from message (3.10), i.e., P matches the expected pre-shared secret, M is authenticated to S. Also, $f(R_S)$ indicates to S that the current protocol run is fresh, as S knows that R_S is freshly generated. S believes that K_{MS} is a fresh encryption key as R_S is freshly generated by S in the current protocol run. S also believes that M knows K_{MS} as M is able to encrypt P with the key. Hence for S the protocol goals have been established, i.e., S learns the authenticated identity of M (i.e., the user U), and S and M share a fresh encryption key K_{MS}. When M receives message (3.11) and verifies $f(R_M)$, M believes the following: the communication involves the authenticated (true) party S, as none other than S can retrieve R_M in message (3.10); that S knows K_{MS}; and that the current protocol run is fresh. Hence the protocol goals for M are also established. Messages (3.9) and (3.10) are cryptographically linked by R_S, and messages (3.10) and (3.11) are cryptographically linked by R_M. This chaining prevents *replay* and *interleaving* attacks [68].

We now make a few additional comments regarding possible alternatives. Apparently $f(R_S)$ could be removed from message (3.10), as verification of P requires use of K_{MS}, thereby indicating freshness of the current protocol run to S; however, verification of P would generally require a database access (where userid and password pairs are stored). Using $f(R_S)$ in message (3.10) enables S to determine freshness of the current protocol run directly from this message; thus the preference to retain $f(R_S)$ in message (3.10). Note that the use of $f(R_M)$ in message (3.11) is intended to allow verification by M that S knows K_{MS}; thus alternately $f(R_M)$ might be replaced by any constant in this message. Also, while calculating $f(R_S)$ and $f(R_M)$, $f(\cdot)$ can be as simple as an identity function instead of a cryptographically secure hash function.

3.3.2 Unsuccessful Attacks Against MP-Auth

We list several potential attacks against MP-Auth, and discuss how MP-Auth prevents them. We also discuss some MP-Auth steps in greater detail, and further motivate various protocol components/steps.

a) Remote desktop attacks. A malicious browser B can collect message (3.4) and then deny access to U. B can use message (3.4) to login to S, and provide an attacker a remote desktop, e.g., a Virtual Network Computing (VNC) terminal, in real-time to the user PC. However, this attack will be detected by the transaction integrity confirmation step of MP-Auth.

b) Session hijacking attacks. In a session hijacking attack, malware may take control of a user session after the user successfully establishes a session with the legitimate server; e.g., B may leak K_{BS} to malware. The malware may actively alter user transactions, or perform unauthorized transactions without immediately being noticed by the user. However, such attacks will be detected by the transaction integrity confirmation step of MP-Auth.

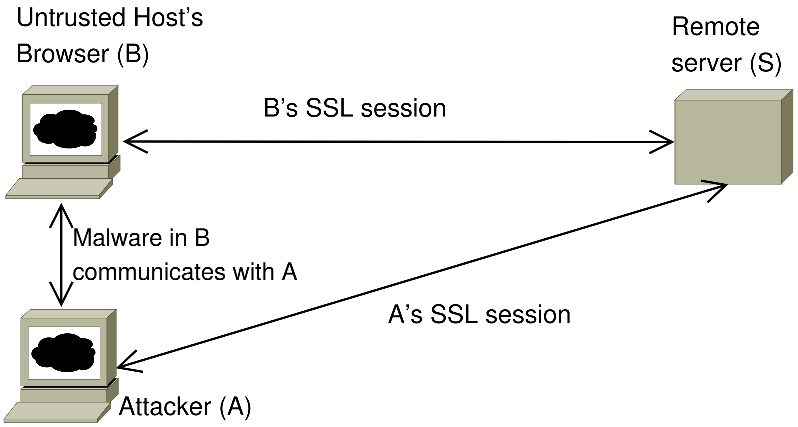


Figure 3.2: Setup for a parallel session attack

c) Parallel session attacks. In a parallel session attack [68], messages from one protocol run are used to form another successful run by running two or more protocol instances simultaneously. Generally a parallel session attack may effectively be prevented through the proper chaining of protocol messages. However, in MP-Auth, there is no authentication between M and B, making such an attack possible even when protocol messages are linked correctly. An attack against MP-Auth is the following (see Fig. 3.2). When U launches B to visit S's site, malware from U's PC notifies a remote attacker A. A starts another session with S as U, and gets message (3.1) from S, which the attacker relays to U's PC. The malware on U's PC drops the message (3.1) intended for U when B attempts to send the message to M, and forwards the attacker's message to M instead. The malware then relays back U's response (i.e., from M) to A. Now A can login as U for the current session, although A is unable to learn P. However, the transaction integrity confirmation step in MP-Auth makes such parallel session attacks meaningless (view-only).

3.3.3 Remaining Attacks Against MP-Auth

Although MP-Auth apparently protects user passwords from malware installed on a PC or phishing websites, here we discuss some other possible attacks against MP-Auth which, if successful, may expose a user's plaintext password.

a) Mobile malware. We have stated the requirement that the personal (mobile) device be trusted. An attack could be launched if attackers can compromise mobile devices, e.g., by installing a (secret) keylogger. Malware in mobile networks is increasing as high-end cellphones (smart phones) contain millions of lines of code. For example, a Sept. 2006 study [101] reported that the number of existing malware for mobile devices is nearly 162; in comparison, the number of viruses has reportedly crossed the one million mark in the PC world in 2007 [57]. Worms such as Cabir [81] are designed to spread in smart phones by exploiting vulnerabilities in embedded operating systems. Regular cellphones which are capable of running J2ME MIDlets have also been targeted, e.g., by the `RedBrowser` Trojan [82]. However, currently cellphones remain far more trustworthy than PCs, thus motivating our proposal.

In the future, as mobile devices increasingly contain much more software, the requirement of trustworthy cellphones becomes more problematic, and their use for sensitive purposes such as online banking makes them a more attractive target. Limited functionality devices (with less software, implying more trustworthy, see e.g., Laurie and Singer [148]) may then provide an option for use with MP-Auth. Even if MP-Auth is implemented in such a special-purpose or lower functionality device (e.g., an

EMV CAP reader[4]), the device can hold several public keys for different services; in contrast, users may require a separate passcode generator for each service they want to access securely in standard two-factor authentication proposals. Another possibility of restricting mobile malware may be the use of micro-kernels [119], formally verifiable OS kernels [266], protections against virtual-machine based rootkits (VMBRs) [142], or a virtualized Trusted Platform Module (vTPM) [251] on cellphones to restore a trustworthy application environment. The Trusted Computing Group's (TCG's) Mobile Phone Work Group (MPWG) is currently developing specifications [168] for securing mobile phones.

In version 9 of the Symbian OS (a widely used cellphone OS), Symbian has introduced *capabilities* and *data caging* [240]. A capability allows access to a set of APIs for an application, which is managed through certification, e.g., Symbian Signed.[5] About 60% of APIs are available to all applications without any capabilities. Some capabilities are granted at installation time by a user. Some sensitive APIs (e.g., `ReadDeviceData`, `TrustedUI`) are granted only after passing Symbian Signed testing. Capabilities such as DRM must be granted by the device manufacturer, e.g., Nokia. Controlled capabilities may restrict functionality of unauthorized applications (or malware). Access to the file system for applications and users is restricted through data caging. Caging enforces data privacy so that an application can access only its private directories and directories marked as *open*.

Enforcement of capabilities and data caging is done by Symbian Trusted Computing Base (TCB). TCB is a collection of software including the kernel, file system, and software installer. However, TCB will become an attractive attack target, and it may contain bugs in itself. Secure hardware, e.g., Trusted Platform Module (TPM) may help achieve goals of Symbian Platform Security.

[4] EMV is a commercial standard (Europay, MasterCard and Visa) for interoperation of chip cards. EMV CAP (Chip Authentication Programme) is a two-factor authentication system for bank customers with chip cards where a card is used to generate a one-time password; see `https://emvcap.com`. A chip card is inserted into a small CAP reader (which includes a small display, keypad and a low-end processor), and using the PIN associated with the card, a user can generate one-time passwords, respond to a server's challenge and MAC over transaction data. See Drimer et al. [71, 72] for recent attacks on such EMV CAP implementations as deployed in in the U.K.

[5] `www.symbiansigned.com`

Anti-virus software (e.g., Trend Micro [264]) for mobile platforms may also help maintain trustworthiness of cellphones. Malware targeting mobile phones is still limited, and leveraging the experience of working to secure traditional PC platforms may help us achieve a relatively secure mobile computing environment. However, considering the current state of mobile phone security, MP-Auth would perform better on devices whose software upgrade is tightly controlled (e.g., only allowing applications which are digitally signed by a trustworthy vendor).

b) Common-password attacks. Users often use the same password for different websites. To exploit such behaviour, in a common-password attack, attackers may break into a low-security website to retrieve userid/password pairs, and then try those in financially critical websites, e.g., for online banking. MP-Auth itself does not address the common-password problem (but see e.g., PwdHash [214]).

c) Social engineering. Some forms of social engineering remain a challenge to MP-Auth (and apparently, other authentication schemes using a mobile device). For example, malware might prompt a user to enter the password directly into an untrusted PC, even though MP-Auth requires users to enter passwords only into a cellphone. In a *mixed* phishing attack,[6] emails are sent instructing users to call a phone number which delivers, by automated voice response, a message that mimics the target bank's own system, and asks callers for account number and PIN. Fraudsters may also exploit transaction integrity confirmation using similar payee names, e.g., Be11 Canada instead of Bell Canada, or modifying a transaction amount e.g., from $100 to $1000. User habit or user instruction may provide limited protection against these. However, we argue that phishing attacks against transaction confirmation may be more easily detected than those attacks against financial sites; apparently users possess better understanding of the consequences of "pay $1000 to party X" than the security cues of a given site (e.g., detecting correct URLs, and comprehending SSL site certificates).

d) Private key disclosure. It would be disastrous if the private key of a bank is compromised. This would require, e.g., that the bank generate and distribute a new public key. However, this threat also exists for currently deployed SSL (server site) certificates, and root keys present in current browsers. If a user has multiple bank

[6] `http://www.cloudmark.com/press/releases/?release=2006-04-25-2`

accounts that use MP-Auth, compromising one of those bank private keys may expose passwords for other accounts. The attack[7] may work in the following way. Assume a user has accounts in banks $S1$ and $S2$ with server IDs ID_{S1} and ID_{S2}, and the private key for $S1$ has been compromised. The user goes to $S2$'s website for online banking. Malware in the user's PC forwards ID_{S1} to the cellphone while displaying $S2$'s website on the PC. If the user inputs the userid and password for her $S2$ account without carefully checking the displayed server ID on her cellphone, the attacker can now access the user's password for $S2$ (using $S1$'s private key). Displaying a distinct image of the requesting server on the cellphone may reduce such risks.

e) Shoulder surfing attacks. A nearby attacker may observe (*shoulder surf*) while a user enters a password to a mobile device. Video recorders or cellphones with a video recording feature can also easily record user passwords/PINs in a public location, e.g., in an ATM booth. MP-Auth does not stop such attackers. Methods resilient against shoulder surfing have been proposed (e.g., [275], [215]), and may be integrated with MP-Auth, although their practical viability remains an open question.

f) Online password guessing. Since MP-Auth assumes passwords as the only shared secret between a user and a server, online password guessing attacks can be launched against MP-Auth. Current techniques, e.g., locking online access to an account after a certain number of failed attempts, can be used to restrict such attacks.

3.4 Usability and Deployment

In this section, we discuss usability and deployment issues related to MP-Auth. Usability is a great concern for any protocol intended to be used by general users, e.g., for Internet banking and ATM transactions. In MP-Auth, users must connect a cellphone to a client PC. This step is more user-friendly when the connection is wireless, e.g., Bluetooth, than wire-line. Then the user browses to a bank website, and enters into the cellphone the userid and password for the site (step 5 in MP-Auth, see Section 3.2). We also assume that typing a userid and password on a cellphone keypad is acceptable in terms of usability, as many users are accustomed to type SMS

[7] An anonymous FC 2007 referee pointed us this attack.

messages or have been trained by BlackBerry/Treo experience. However, verification of transactions may be challenging to some users. We have not conducted any user study to this end.

During authentication the cryptographic operations a cellphone is required to perform in MP-Auth include: one public key encryption, one symmetric encryption and one decryption, one random number generation, and three cryptographic hash operations. The most expensive is the one public key encryption, which is a relatively cheap RSA encryption with short public exponent in our application; see Section 3.5 for concrete results.

For authentication in MP-Auth, a bank server performs the following operations: one private key decryption, one symmetric key encryption and one decryption, three cryptographic hash operations, and one random number generation. The private key decryption will mostly contribute to the increment of the server's computational cost. Verification of one-time passcodes generated by hardware tokens or SMS passcodes (as deployed in many two-factor authentication schemes) also incurs extra processing and infrastructure costs. However, currently we are unable to compare the costs of MP-Auth with that of existing two-factor techniques due to unavailability of such data.

Banks may also hesitate in distributing software programs for a user's PC and mobile device as required by MP-Auth, apparently due to software maintenance issues. Standardization of such software APIs might enable interoperable independent tools development, and thus reduce maintenance burdens. If a specialized device like the EMV CAP reader is used for MP-Auth, banks may pre-package all require software on the device and relieve users from installing anything on a personal device. However, users and banks may still need to deal with software for communications between a PC and personal device.

We now discuss other usability and deployment aspects which may favour MP-Auth (see also Section 4.1).

1. As it appears from the current trend in online banking (see Section 4.1.1), users are increasingly required to use two-factor authentication (e.g., with a separate device such as a SecurID passcode generator) for login. Hence using an existing mobile device for online banking relieves users from carrying an

extra device. Also, a user might otherwise require multiple hardware tokens (e.g., SecurID, Chip and PIN card) for accessing different online accounts (from different banks).

2. MP-Auth offers cost efficiency for banks – avoiding the cost of providing users with hardware tokens (as well as the token maintenance cost). The software modification at the server-end is relatively minor; available SSL infrastructure is used with only three extra messages (between a browser and server) beyond SSL. MP-Auth is also compatible with the common SSL setup, i.e., a server and a client authenticate each other using a third-party-signed certificate and a user-chosen password respectively.

3. Several authentication schemes involving a mobile device store long-term secrets on the device. Losing such a device may pose substantial risk to users. In contrast, losing a user's cellphone is inconsequential to MP-Auth assuming no *secret* (e.g., no "remembered password") is stored on the phone.

4. Public key distribution and renewal challenges usability in any PKI. Key updating is also troublesome for banks. However, key renewal is an infrequent event; we assume that users and banks can cope with this process once every two to three years. If key updates are performed through the mobile network or selected ATMs (e.g., within branch premises), the burden of key renewal is largely distributed. For comparison, hardware tokens (e.g., SecurID) must be replaced approximately every two to five years.

5. One usability problem of MP-Auth is that users must deal with two devices (a trusted device and a PC) for online banking. Since usability of smartphones is increasing with the adoption of a full `QWERTY` keyboard and a relatively large (e.g., 320 x 240 pixel) colour screen, it would be better if MP-Auth could be used directly from such a device (i.e., without requiring a PC). However, we do not recommend such an integration as it may be vulnerable to phishing attacks when a phishing website mimics MP-Auth's user-interface for password input.

Although we have not tested MP-Auth for usability, the above suggests that compare to available two-factor authentication methods (see Section 4.1.1), MP-Auth may be as usable or better. However, we hesitate to make strong statements without usability tests (cf. [51]).

3.5 Implementation and Performance

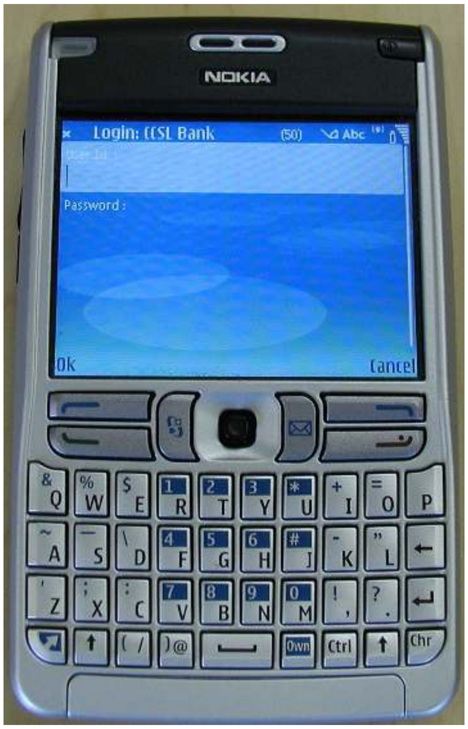


Figure 3.3: MP-Auth login

We developed a prototype of the main authentication and session key establishment parts of MP-Auth to evaluate its performance. Our prototype consists of a web server, a Firefox extension, a desktop client, and a MIDlet application on the cellphone. We set up a test web server (bank), and used PHP `OpenSSL` functions and the `mcrypt` module for the server-side cryptographic operations. The Firefox extension communicates between the web server and desktop client. The desktop client forwards messages to and from the cellphone over Bluetooth. We did not have to modify the web server or Firefox browser for MP-Auth besides adding PHP scripts to the login page (note that Phoolproof [195] requires browser code modifications). We used the `BlueZ` Bluetooth protocol stack for Linux, and Rococosoft's Impronto Developer Kit for Java. We developed a MIDlet – a Java application for Java 2 Micro Edition (J2ME), based on the Mobile Information Device Profile (MIDP) specification – for a Nokia E62 phone (commercially available circa Sept. 2006, running Symbian Series 60 r3 on a Texas Instruments processor at 235 MHz); see Fig. 3.3. For cryptographic operations on the MIDlet, we used the Bouncy Castle Lightweight Crypto API.

To measure login performance, we used MP-Auth for over 200 successful logins, and recorded the required login time, i.e., the time to complete steps 1 through 9 in (see Section 3.2; excluding userid and password input in step 5). We carried out similar tests for regular SSL logins. The results are summarized in Table 3.2.

	Avg. Time (s)	[Min, Max] (s)
MP-Auth	0.62	[0.34, 2.28]
Regular SSL	0.08	[0.06, 0.22]

Table 3.2: Performance comparison between MP-Auth and regular SSL login excluding user input time (in seconds)

Public key encryption	`RSA` 1024-bit
Symmetric encryption	`AES-128` (CBC)
Hash function	`SHA-1` (160-bit)
Source of randomness	`/dev/urandom`, `SecureRandom`

Table 3.3: Cryptosystems and parameters for MP-Auth

Table 3.3 summarizes other implementation details. Although regular SSL login is almost eight times faster than MP-Auth, on average, it takes less than a second for MP-Auth login. We believe that this added delay would be acceptable, given that entering a userid and password takes substantial additional time.

3.6 Concluding Remarks

We have proposed MP-Auth, a protocol for web authentication which is resilient to keyloggers (and other malware including rootkits), phishing websites, and session hijacking. Recently, many new small-scale, little-known malware instances have been observed that install malicious software launching keylogging and phishing attacks; these are in contrast to large-scale, high-profile worms like Slammer. One reason for this trend might be the fact that attackers are increasingly targeting online financial transactions.[8] Furthermore, such attacks are fairly easy to launch; for example, attackers can gain access to a user's bank account simply by installing (remotely) a keylogger on a user PC and collecting the user's banking access information (userid and password). MP-Auth is designed to prevent such attacks. MP-Auth primarily focuses on online banking but can be used for general web authentication systems as well as at ATMs. Our requirement for a trustworthy personal device (i.e., free of malware) is important, and becomes more challenging over time, but as discussed in Section 3.3.3, may well remain viable. In our MP-Auth implementation, cryptographic computations and Bluetooth communications took less than a second for login (excluding the user input time), which we believe to be an acceptable delay for the added security. Despite a main objective of preventing phishing and keylogging

[8] According to one report [13], 94.2% of all phishing sites in Feb. 2008 targeted online financial services, e.g., online banking and credit card transactions.

attacks, MP-Auth as presented remains one-factor authentication; thus an attacker who nonetheless learns a user's password can impersonate that user. Consequently, the server side of MP-Auth must be trusted to be secure both against insider attacks and break-ins. We revisit these attacks in Section 4.2.

Users often input reusable critical identity information to a PC other than userid and password, e.g., a passport number, social security number, driver's licence number, or credit card number. Such identity credentials are short, making them feasible (albeit tedious) to enter from a cellphone keypad. In addition to protecting a user's userid/password, MP-Auth may easily be extended to protect other identity credentials from the reach of online attackers, and thereby might be of use to reduce online identity theft. We believe that the very simple approach on which MP-Auth is based – using a cellphone or similar device to asymmetrically encrypt passwords and one-time challenges – is of independent interest for use in many other applications, e.g., traditional telephone banking directly from a cellphone, where currently PINs are commonly transmitted in-band without encryption.

Chapter 4

MP-Auth: Background and Enhancements

We introduce the Mobile Password Authentication (MP-Auth) protocol in Chapter 3. In this chapter, we discuss related work to MP-Auth, including commercial one-time password generators, and a number of web authentication techniques proposed in the literature. We also discuss two proposals complementary to MP-Auth.

4.1 Survey of MP-Auth Related Work

In this section, we summarize and provide extended discussion of related online authentication methods used in practice or proposed in the literature, and compare MP-Auth with these techniques.

4.1.1 Online Authentication Methods

We first discuss several online authentication methods commonly used (or proposed for use) by banks, and briefly discuss their security.

a) Password-only authentication. Most bank websites authenticate customers using only a password over an SSL connection. This is susceptible to keyloggers and phishing. Banks' reliance on SSL certificates does not stop attackers. Attackers have used certificates – both self-signed, and real third-party signed certificates for *sound-alike* domains, e.g., `visa-secure.com` – to display the SSL lock on phishing websites. In 2005, over 450 phishing websites were reported to deploy SSL [181]. A trojan (with rootkit capabilities) has been reported [89] to inject a spoofed HTML form (from the local PC) inside a SSL-protected bank website for collecting banking and identity information; the form appears to be served by the real bank site, and the browser displays the correct SSL site certificate when verified by the user. Also, the bank site remains uncompromised in this attack.

In a cross-site/cross-frame scripting attack, vulnerable website software is exploited to display malicious (phishing) contents within the website, making such attacks almost transparent to users. Past vulnerable websites include Charter One Bank, MasterCard, Barclays and Natwest [294]. In a March 2006 phishing attack, attackers broke into web servers of three Florida-based banks, and redirected the banks' customers to phishing websites.[1] In another high-profile phishing attack, attackers manipulated a U.S. government website to forward users to phishing websites.[2]

Reliance on SSL itself also leads to problems. For example, only one in 300 customers of a New Zealand bank [181] chose to abandon the SSL session upon a browser warning indicating an expired SSL site certificate; the bank accidentally allowed a certificate to expire for a period of 12 hours. A user study by Dhamija et al. [67] also notes that standard (visual) security indicators on websites are ineffective for a significant portion of users; over 90% participants were fooled by phishing websites in the study.

As front-end (client-side) phishing solutions are failing in many instances, some banks are putting more resources at back-end fraud detection to counter phishing threats. For example, HSBC in Brazil uses the `PhishingNet`[3] back-end solution from The 41st Parameter. PhishingNet uses user machine identification, and monitors online account activities, without requiring any user registration or software downloads. Such a solution is almost transparent to end-users, and may help detect fraudulent transactions. The RSA Adaptive Authentication for Web[4] also provides similar back-end fraud detection capabilities. However, in case of session hijacking attacks when fraudulent transactions are performed from a user's own machine, back-end solutions may not help much.

The above suggests password-only web authentication over SSL is inadequate in today's Internet environment. This is motivating financial organizations towards two-factor authentication methods.

[1] http://news.netcraft.com/archives/2006/03/27/phishers_hack_bank_sites_redirect_customers.html

[2] http://www.eweek.com/article2/0,1895,1894746,00.asp

[3] http://www.the41.com/site/solutions_phishing.html

[4] http://www.rsa.com/node.aspx?id=3018

b) Two-factor authentication. Traditionally, authentication schemes have relied on one or more of three factors: something a user *knows* (e.g., a password), something a user *has* (e.g., a bank card), and something a user *is* (e.g., biometric characteristics). Properly designed authentication schemes that depend on more than one factor are more reliable than single-factor schemes. Note that the authentication scheme used in ATMs through a bank card and PIN is two-factor; but, an online banking authentication scheme that requires a user's bank card number (not necessarily the card itself) and a password is single-factor, i.e., both are something *known.* As a step toward multi-factor authentication, banks are providing users with devices like one-time password generators, to use along with passwords for online banking, thus making the authentication scheme rely on two independent factors. Examples of two-factor authentication in practice are given below.

1. Several European banks attempt to secure online banking through e.g., standalone, offline passcode generators.[5]

2. U.S. federal regulators have provided guidelines for banks to implement two-factor authentication by the end of 2006 for online banking [83].

3. The Association of Payment and Clearing Systems (APACS) in the U.K. is developing a standard[6] for online and telephone banking authentication. Most major U.K. banks and credit-card companies are members of APACS. The standard provides users a device to generate one-time passwords using a chip card and PIN. The one-time password is used along with a user's regular password.

Two-factor web authentication methods may make the collection of passwords less useful to attackers and thus help restrict phishing attacks. However, these methods raise deployment and usability issues, e.g., cost of the token, and requirement to carry the token. Also malware on a client PC can record the device-generated secret (which a user inputs directly to a browser), and log on to the bank website before the actual user. This is recognized as a classic man-in-the-middle (MITM) attack [234]. Apparently showing a pre-selected user image or phrase on the login page, or "device

[5] For a list of bank sites that use SSL login and/or two-factor authentication see `https://www.securewebbank.com/loginssluse.html`.

[6] `http://www.chipandpin.co.uk/`

fingerprinting" (information specific to a user PC, e.g., client browser, OS, CPU type, and screen resolution) are considered as a second factor by several U.S. banks; see O'Connor [190] for how easily these second factors can be defeated by traditional phishing/MITM attacks.

In an interesting real attack [260] against a one-time password scheme implemented by a Finnish bank, the bank provided users a scratch sheet containing a certain number of one-time passwords. By setting up several phishing sites, attackers persuaded users to give out a sequence of one-time passwords in addition to their regular passwords. This attack is made more difficult if one-time passwords expire after a short while (e.g., 30 to 60 seconds in SecurID); then the collected one-time passwords must be used within a brief period of time from a user's login attempt. A July 2006 phishing attack [180], attackers collected userid, password, as well as one-time password (OTP) generated by time-based passcode generators from Citibank customers, and launched a real-time MITM attack against compromised accounts. Also, such time-based passcode generators, e.g., SecurID, typically have time synchronization problems between a client device and the server [289], and expire in 2-5 years. Other security issues of such devices (e.g., [279], [173], [34], [198]) are not directly relevant to our discussion; we assume that any weaknesses could be repaired by superior algorithms or implementations overtime, albeit with the usual practical challenges, e.g., backwards compatibility.

Note that, even when a one-time password is used along with a user's (long-term) regular password, gathering long-term passwords may be still be of offline use to an attacker. For example, if flaws are found in a one-time key generator algorithm (e.g., *differential adaptive chosen plaintext* attack [34]) by which attackers can generate one-time keys without getting hold of the hardware token, keylogging attacks to collect user passwords appear very useful.

Instead of gathering passwords, attackers can simply steal money from user accounts in real-time, immediately after a user completes authentication [165, 130, 255]. Therefore, transaction security becomes critical to restrict such attacks.

c) Transaction security and complimentary mechanisms. To protect important transactions, and make users better able to detect break-ins to their accounts, some banks have deployed security techniques which are generally complementary to authentication schemes. Examples include:

1. Two New Zealand banks require online users to enter a secret from a cellphone (sent as an SMS message to the phone) for transfers over $2500 from one account to another [259]. Bank of America also offers a similar feature called SafePass[7] for authorizing new payees and online money transfers, and allowing higher transfer limits. The passcode is sent to a user's cellphone, or can be generated from a wallet-sized stand-alone card.

2. Customers of the Commonwealth Bank of Australia[8] must answer (pre-established) identification questions when performing sensitive transactions. Email alerts are sent to users to confirm when users' personal details have been changed, or modifications to user accounts are made.

3. Bank of America uses SiteKey[9] to strengthen online authentication. If a user PC is recognized by the bank, a secret pre-shared SiteKey picture is displayed; upon successful verification of the SiteKey picture, the user enters her password. A confirmation question is asked if the user PC is not recognized, and the SiteKey picture is displayed when the user answers the question correctly. The SiteKey picture provides evidence that the user is entering her password to the correct website.

In principle, the above mechanisms (as well as MP-Auth's transaction integrity confirmation) are similar to integrity cross-checks by a second channel [270]. These appear to be effective only against high-impact online frauds. Attackers may be able to defeat some of these techniques; e.g., if a bank requires SMS verification on large transactions, attackers can commit several relatively small transactions (e.g., $10 instead of $1000) to avoid the verification step. Also, SMS verification requires access to cellphone networks which is a problem when a phone network is not available (e.g., while traveling).

[7] `http://www.bankofamerica.com/privacy/index.cfm?template=learn_about_safepass`

[8] `http://www.commbank.com.au/Netbank/faq/security.asp`

[9] `http://www.bankofamerica.com/privacy/passmark/`

d) Using a cellphone alone for important Internet services. Some [107] believe cellphones have the potential to replace commodity PCs entirely. One proposed solution to keyloggers is to perform all critical work through a cellphone browser, or through a PDA. However, a combination of the following usability and security issues may restrict such proposals being widely deployed.

1. The display area of a cellphone/PDA is much smaller than a PC, limiting usability for web browsing.

2. Users may still reveal passwords to phishing sites controlled by malicious parties (through e.g., domain name hijacking [124], Kaminsky DNS-flaw [137]). Thus even a trusted browser in a trusted device may not stop phishing attacks; i.e., such a setup may allow a *secure* pipe directly to phishing sites.

3. In many parts of the world, airtime costs money. So Internet browsing through a mobile network remains, at least presently, far more expensive than wire-line Internet connections.

e) Comparing MP-Auth with existing online authentication methods. In contrast to two-factor authentication methods, by design MP-Auth does not provide attackers any window of opportunity when authentication messages (i.e., collected regular and one-time passwords of a user) can be replayed to login as the legitimate user and perform transactions on the user's behalf. The key observation is that, through a simple challenge-response, message (3.4) in MP-Auth (Section 3.2) effectively turns a user's long-term static password into a one-time password in such a way that long-term passwords are not revealed to phishing websites, or keyloggers on an untrusted PC. In contrast to transaction security mechanisms, MP-Auth can protect both large and small transactions as long as users diligently check integrity confirmation messages, and transactions are prudently labled for user confirmation from the device; for example, even small transactions to an unknown/unregistered party should be categorized as sensitive. Also, MP-Auth does not require text or voice communications airtime for web authentication or transaction security. (See also Section 3.4 for more comparison on usability and deployment issues.)

4.1.2 Academic Proposals

Here we summarize selected academic proposals for authentication from an untrusted PC using a mobile device. MP-Auth shares several design goals with these, and is influenced by the ideas and experiences of these past proposals. We also compare MP-Auth to these in terms of technical merits and usability.

a) Splitting trust paradigm. Abadi et al. [1] envisioned an *ideal* smart-card (with an independent keyboard, display, processor) as early as 1990, and designed protocols using such a device to safeguard a user's long-term secrets from a potentially malicious computer. In 1999, Balfanz and Felten [20] proposed a scheme to deliver smart card functionality through a PalmPilot assuming the availability of user-level public key systems. They introduced the *splitting trust* paradigm to split an application between a small (in size and processing power) trusted device and an untrusted computer. Our work is based on such a paradigm where we provide the long-term password input through widely available cellphones, and use the untrusted computer for computationally intensive processing and display. However, we do not use any user-level PKI.

b) Phoolproof phishing prevention. Parno, Kuo and Perrig [195] proposed a cellphone-based technique to protect users against phishing with less reliance on users making *secure* decisions. With the help of a pre-shared secret – established using an out-of-band channel, e.g., postal mail – a user sets up an account at the intended service's website. The user's cellphone generates a key pair $\{K_U, K_U^{-1}\}$, and sends the public key to the server. The user's private key and server certificate are stored on a cellphone for logins afterward. During login (see Fig. 4.1), a user provides userid and password to a website on a browser (as usual), while in the background, the browser and server authenticate (using SSL mutual authentication) through the pre-established client/server public keys in an SSL session; the browser receives the client public key from the cellphone. (See also the Personal Transaction Protocol (PTP) [167] for a similar approach from leading mobile phone manufacturers.)

In Figure 4.1, DH_S, DH_C represent the Diffie-Hellman public key parameters for the server and client browser respectively, and h is a secure hash of all previous SSL handshake messages of the current session. As noted [195], attackers may hijack

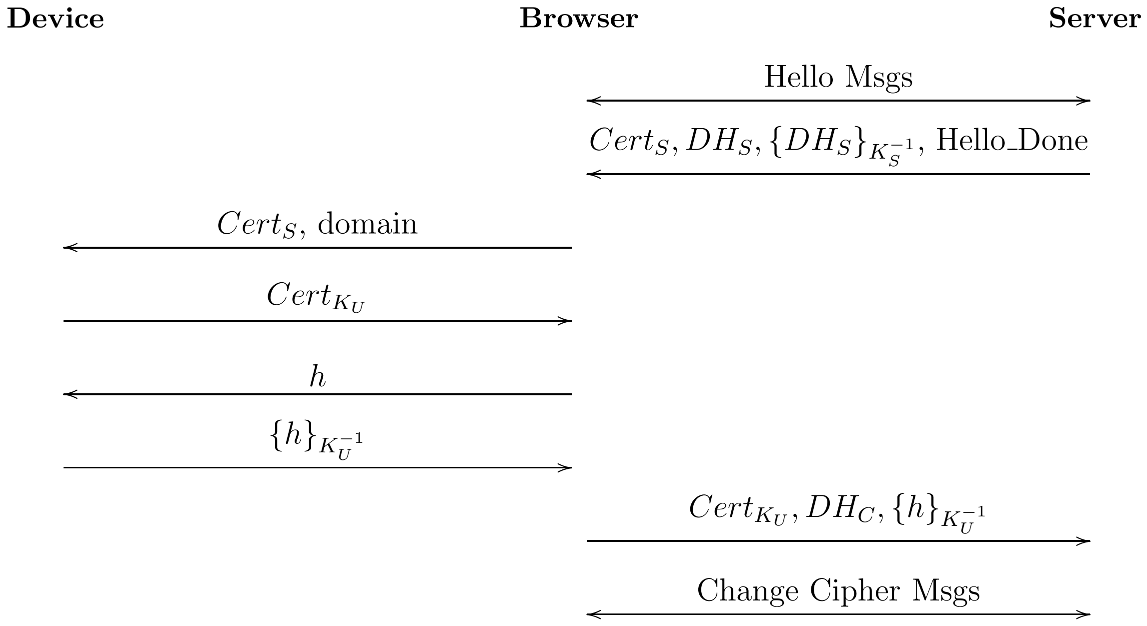


Figure 4.1: Phoolproof login process (adapted from [195])

account setup or (user) public key re-establishment. Phoolproof assumes that users can correctly identify websites at which they want to set up an account. Public key creation in Phoolproof happens in the background and is almost transparent to users. However, users must *revoke* public/private key pairs in case of lost or malfunctioning cellphones, or a replacement of older cellphone models. Expecting non-technical users (e.g., typical bank customers) to understand concepts of revocation and renewal of public keys may not be practical yet.

It is also assumed in Phoolproof that the (Bluetooth) channel between a browser and cellphone is secure. Seeing-is-believing (SiB) [159] techniques are proposed to secure local Bluetooth channels, requiring users to take snapshots using a cameraphone, apparently increasing complexity to users. If malware on a PC can replace h (when the browser attempts to send h to the cellphone) with an h value from an attacker, the attacker can login as the user (recall *Parallel Session Attacks* in Section 3.3.2). Also, Phoolproof is not designed to provide protection against session hijacking attacks, which are becoming more common, and easy to develop and deploy [130]. MP-Auth achieves such protection at the (human interaction) cost of transaction integrity confirmation.

c) Bump in the Ether. Bump in the Ether (BitE) [160] proxies sensitive user input to a particular application via a trusted mobile device, bypassing the Linux X-windowing system. Users receive verifiable evidence regarding the integrity of the host kernel and whether the intended user application has been loaded. Only the target application receives user input from the mobile device through a user-verifiable trusted tunnel (between the device and application). BitE assumes the OS kernel is trustworthy, and the BIOS and OS are TPM-enabled and perform integrity check of code loaded for execution. BitE requires a user's mobile device to be cryptographically paired with the OS kernel. For establishing a trusted tunnel between the user device and an application, (symmetric) cryptographic keys for each BitE-aware application must be shared beforehand (e.g., during application installation/registration). Keystrokes from the trusted device is then sent encrypted from the device to the target application.

BitE can protect user input against user-space malware. However, BitE does not protect user inputs from a phishing website, or compromised (e.g., *Trojaned*) user applications. BitE also stores cryptographic keys to the mobile device, which are subject to compromise if the device is lost or left unattended (if not protected otherwise, e.g., through TPM). Session hijacking is also not addressed by BitE.

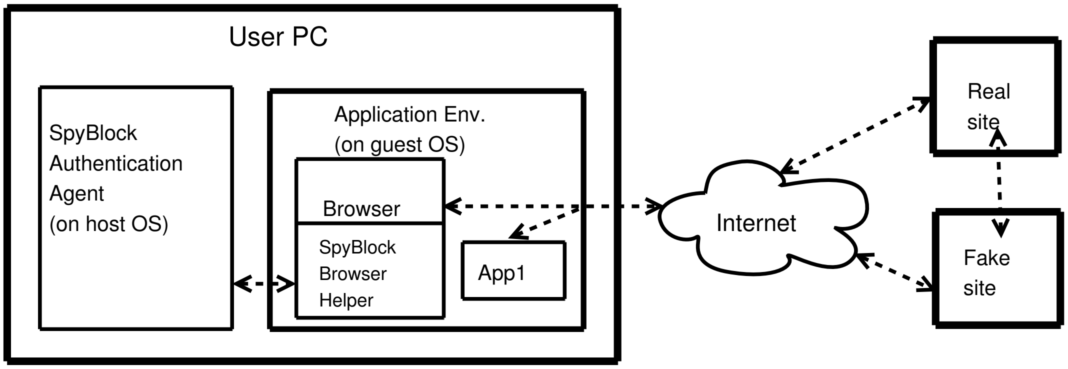

User PC
SpyBlock Authentication Agent (on host OS)
Application Env. (on guest OS)
Browser
SpyBlock Browser Helper
App1
Internet
Real site
Fake site

Figure 4.2: SpyBlock setup (adapted from [129])

d) SpyBlock. Jackson, Boneh and Mitchell propose SpyBlock [129] (see also [130]) to provide *spyware-resistant* web authentication using a virtual machine monitor (VMM). The SpyBlock authentication agent runs on a host OS (assumed to be trusted), and user applications including a web browser with a SpyBlock browser helper run inside an untrusted guest VM on the trusted host OS. See Figure 4.2.

A user authenticates to a website with the help of the SpyBlock authentication agent. The site password is given only to the authentication agent which supports several authentication techniques, e.g., password hashing, strong password authentication, transaction integrity confirmation. The authentication agent provides a *trusted* path to the user through a pre-shared secret picture.

SpyBlock does not require an additional hardware device (e.g., a cellphone). However, a VMM must be installed; the current reality is that most users do not use any VMM. Also, users must know when they are communicating with the authentication agent; user interface design in such a setting appears quite challenging. Another assumption in SpyBlock is that the host OS is *trusted*. In reality, maintaining trustworthiness of any current consumer OS is very difficult (which is in part why secure web authentication is so complex).

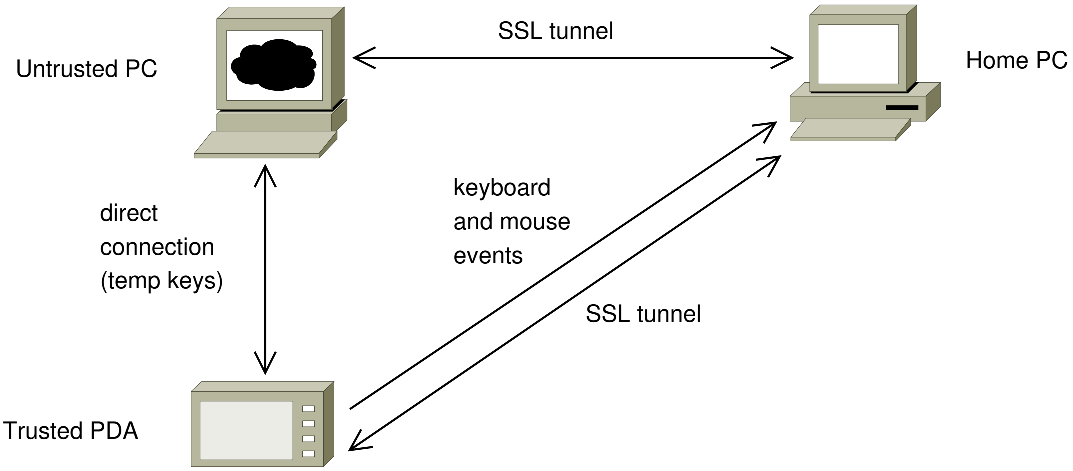


Figure 4.3: Three-party VNC protocol (adapted from [193])

e) Three-party secure remote terminal protocol. Oprea et al. [193] proposed a three-party protocol (see Fig. 4.3) to provide secure access to a home computer from an untrusted public terminal. A trusted device (e.g., PDA) is used to delegate temporary credentials of a user to an untrusted public computer, without revealing any long-term secret to the untrusted terminal. Two SSL connections are established in the protocol: one from the trusted PDA and another from the untrusted terminal to the home PC using a modified Virtual Network Computing (VNC) system. The PDA authenticates normally (using a password) to the home PC, and forwards temporary secret keys to the untrusted terminal. A user can control how much information from the home PC is displayed to the untrusted PC. Control messages to the home VNC, e.g., mouse and keyboard events, are only sent from the PDA.

This protocol safeguards user passwords only when users access a PC (or application) that they control, e.g., a home PC. Also, the trusted device must have SSL capabilities, and is required to maintain a separate SSL channel from the PDA to the home PC.

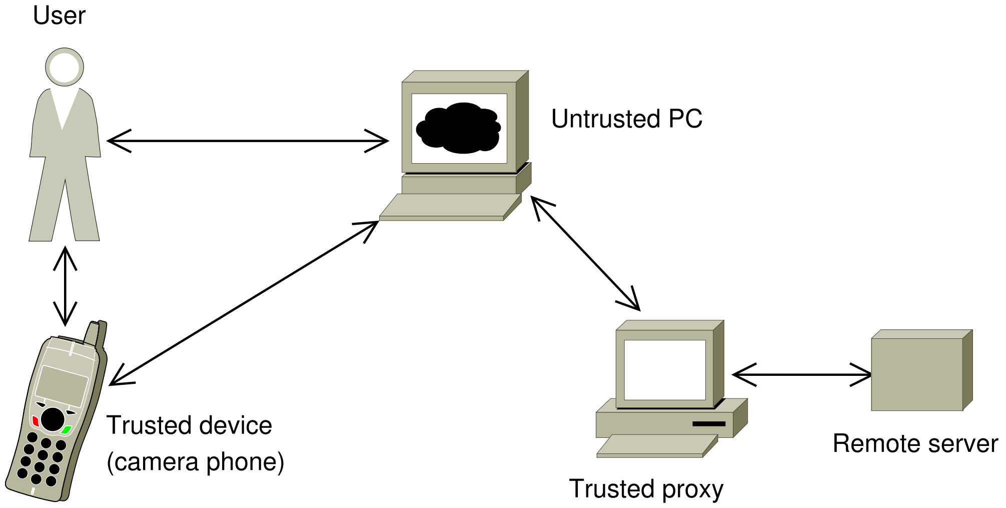


Figure 4.4: Camera-based authentication

f) Camera-based authentication. Clarke et al. [53] proposed a technique using camera-phones for authenticating visual information (forwarded by a trusted service) in an untrusted PC. This method verifies message authenticity and integrity for an entire user session; i.e., it authenticates the content displayed on a PC screen for every webpage or only critical pages in a user session. A small area on the bottom of a PC screen is used to transmit security parameters (e.g., a nonce, a one-time password, or a MAC) as an image, with a strip of random-looking data. Figure 4.4 outlines the proposed protocol.

To access a service from the Internet through an untrusted PC, this scheme requires a *trusted proxy.* A user's long-term keys are stored on the camera-phone, protected by a PIN or biometric measurement. With a stolen phone, an attacker may successfully impersonate the user or retrieve the stored long-term keys from the phone. Camera-based authentication also creates a much different user experience: users are expected to take snapshots and visually verify (cross-check) images in terms of colours and shades. A calibration phase may also be required to construct a mapping between PC screen pixels and camera pixels (in one implementation, reported to take about 10 seconds). It attempts to authenticate contents of a visual display, which is apparently useful in a sense that we can verify what is displayed on the screen.

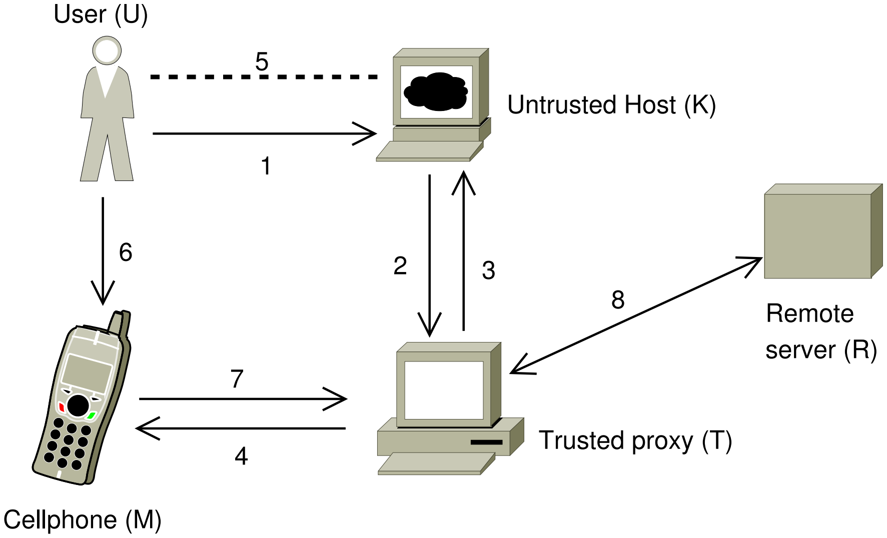


Figure 4.5: Web authentication with a cellphone

g) Secure web authentication with cellphones. Figure 4.5 shows the secure web authentication proposed by Wu et al. [286]. User credentials (userid, password, mobile number etc.) are stored on a trusted proxy server. The protocol involves the following steps (see Fig. 4.5 for symbol definitions).

1. U launches a web browser at K, and goes to T's site.
2. U types her userid and K sends it to T.
3. T chooses a random session name, and sends it to K.
4. T sends this session name to M as an SMS message.
5. U checks the displayed session name at K.
6. U verifies the session name at M.

7. If session names match, the user accepts the session.

8. If U accepts the session, then T uses U's stored credentials to login to R, and works as a web proxy.

This protocol requires a *trusted proxy*, which if compromised, may readily expose user credentials to attackers. A well-behaved proxy may also be tricked to access a service on behalf of a user. Hence the proxy may become a prime target of attacks. Also, losing the cellphone is problematic, as anyone can access the trusted proxy using the phone, at least temporarily. Delegate [133] is another similar trusted proxy based solution for secure website access from an untrusted PC which also provides protection against session hijacking.

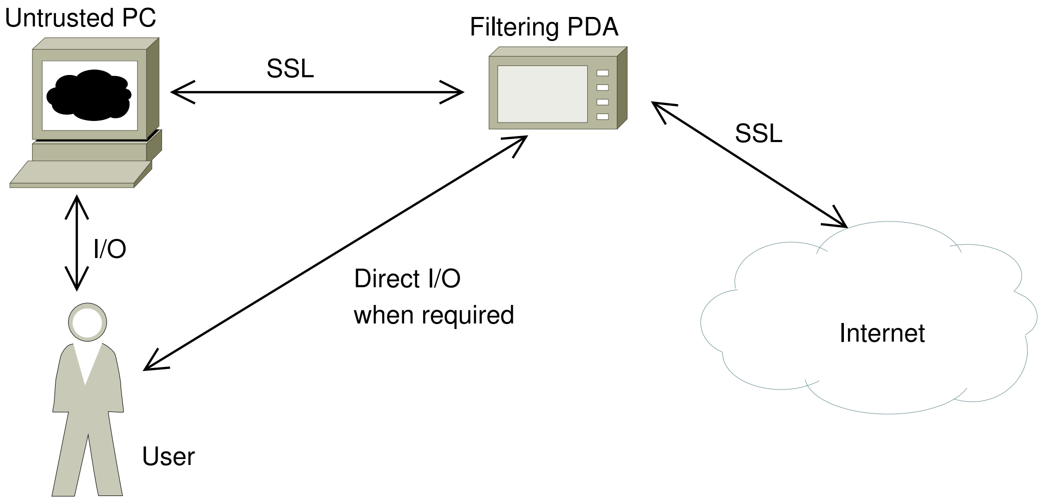


Figure 4.6: Guardian setup (adapted from [156])

h) Guardian: a framework for privacy control. The Guardian [156] framework has been designed with an elaborate threat model in mind. Its focus is to protect privacy of a mobile user,[10] including securing long-term user passwords and protecting sensitive information, e.g., personal data from being recorded (to prevent identity profiling). Guardian works as a personal firewall but placed on a trusted PDA. In effect, the PDA acts as a *portable privacy proxy.* See Figure 4.6.

[10] A user who uses several different public terminals to access critical online services, e.g., banking.

Guardian keeps passwords and other privacy sensitive information out of the reach of keyloggers and other malware installed on an untrusted PC. However, phishing attacks still may succeed. Guardian attempts to manage a large set of sensitive user details, e.g., PKI certificates, SSL connections, and cookies as well as real-time content filtering. Thus its implementation appears to be complex, and requires intelligent processing from the PDA.

i) Comparing MP-Auth with existing literature. Table 4.1 summarizes a comparison of MP-Auth with several academic proposals. An (✗) means a special requirement is needed. An empty box indicates the stated protection is not provided (first three columns), or the stated requirement is not needed (last four columns). NA denotes non-applicability. (All ✓and no ✗ would be optimal.) For example, Phoolproof [195] provides protection against phishing and keylogging, but it is vulnerable to session hijacking; it requires a malware-free mobile and stores long-term secrets on the mobile, but does not require a trusted proxy or trusted PC OS. We acknowledge that although this table may provide useful high-level overview, this does not depict an apple-to-apple comparison. Several solutions listed here require a trusted proxy, thus introduce an extra deployment burden, and present an attractive target to determined attackers. Also, fraudsters may increasingly target mobile devices if long-term secrets are stored on them.

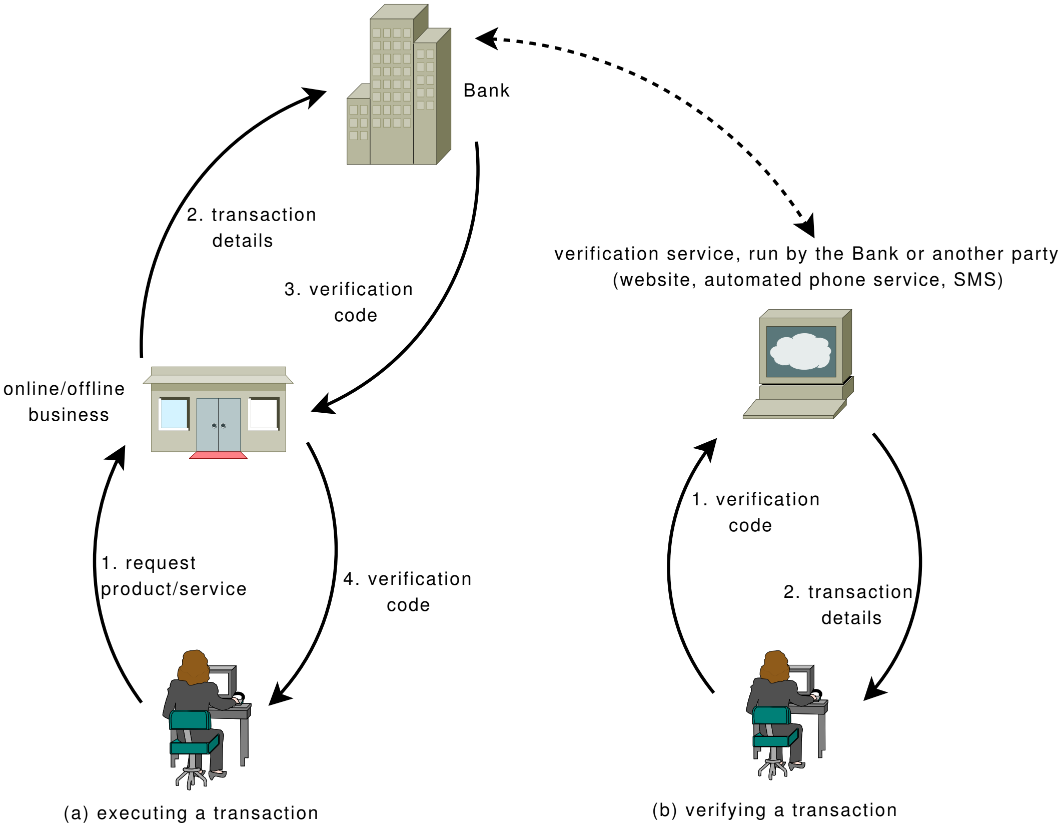
	Protection against			Requirement			
	Session-hijacking	Phishing	Key-logging	Trusted proxy	On-device secret	Trusted PC OS	Malware-free mobile
MP-Auth	✓	✓	✓				✗
Phoolproof [195]		✓	✓		✗		✗
BitE [160]			✓		✗	✗	✗
SpyBlock [129]	✓	✓	✓		NA	✗	
Three-party [193]	NA	NA	✓		✗		✗
Camera-based [53]	✓	✓	✓	✗	✗		✗
Web-Auth [286]		✓	✓	✗	✗		✗
Guardian [156]			✓		✗		✗

Table 4.1: Comparing MP-Auth with existing academic proposals

4.2 Integrity Verification for Financial Transactions

In Chapter 3, we introduced MP-Auth which appears to be a simpler but more effective solution for web authentication and transaction security than other available techniques. However, if a user's password is compromised, MP-Auth cannot prevent or even detect unauthorized transactions, and as currently implemented cannot directly be used for more general web transactions such as credit card transactions.

Below we discuss two preliminary non-cryptographic proposals complementary to MP-Auth, for protecting integrity of online, on-site, and phone/fax/email transactions. Financial organizations in different parts of the world currently employ several innovative techniques to prevent or detect transaction fraud. Our proposals combine and take advantage of those and several academic proposals from the recent past.



(a) executing a transaction

(b) verifying a transaction

Figure 4.7: Executing and verifying a transaction

4.2.1 Verification through a Second Channel

Executing a transaction in this model remains largely the same as the current practice. The difference is that for each transaction (involving a credit or debit card), the corresponding financial institution provides a unique verification code V (similar to tracking numbers used by postal services); see Figure 4.7(a). V should be of sufficient length (e.g., 20 digits/characters long) to prevent collision and guessing attacks. For online transactions, V can be displayed at the end of a transaction; for on-site transactions, V can be printed on the receipt; and for phone/fax/email transactions, V can be transmitted through the same media used for a given transaction. V may also be displayed (on a browser or in a printed receipt) using *QR codes*; these are two-dimensional bar codes, also known as quick response codes, specifically popular in Asian nations such as Japan; see e.g., Figure 4.8 for the QR code representation of the string "FC2009BARBADOS" (generated through `http://qrcode.kaywa.com/`). Users can take a snapshot of a QR code using a camera phone, and decode V from the QR code (requires QR code reader software).

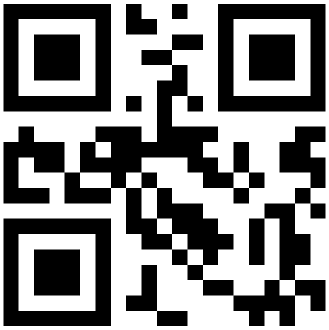


Figure 4.8: QR code for "FC2009BARBADOS"

For verifying a transaction, an independent channel of communication is used, which may include the following. Using V at a *verification* website (run by the user's bank, or the online/on-site merchant), users can retrieve the corresponding transaction details (T). Users may also call an automated toll-free phone service provided by the bank or merchant to get T. Similarly, users can simply send V as an SMS to their bank and get T as a response SMS (or in multiple messages). If QR code is used, users can take a snapshot of the code, convert it to V, and send V as an SMS message, or verify V through the camera-phone's web interface (at a designated/implied website). Emailing V to a pre-specified address may also allow retrieval of T. See Figure 4.7(b). If a discrepancy is found, users can notify their financial institution or the merchant, and resolve the issue as appropriate.

Security of this technique relies on the following assumptions: (i) integrity of the verification service (website/phone/SMS) must be maintained, i.e., attackers cannot insert, remove, or modify transaction details or verification code at the verification service provider or bank sites; (ii) verification codes must be updated immediately after each transaction; (iii) users are aware of the verification service location (e.g., URL/phone number) and can freely and instantly access the service; and (iv) users *carefully* check the detail of a given transaction. Users may choose to verify only a subset of all transactions, e.g., high-value transactions, and transactions executed at less known websites/merchants.

The "security by integrity" paradigm [270] has been proposed for providing data origin assurance through public corroboration (i.e., by cross-checking with information posted at a known or implied website). As an example, if the recipient B of an email wants to verify the integrity of the received email from sender A, B computes a fingerprint/digest (e.g., SHA-1 hash) of the received message, and then verifies the fingerprint at a publicly available *trusted* verification website. Of course, A must publish fingerprints of all emails at the same site (A also must control access to that site). Our proposed scheme is similar in spirit, although we do not require verification codes (corresponding to message fingerprints) to be publicly listed online. Also, we argue that providing means for integrity verification through traditional channels (e.g., voice message) is critical for usability reasons. We require retrieval of transaction details from the verification service; in contrast, message authentication through corroboration [270] requires posting only the fingerprint of a message.

Another related commercial implementation for online banking transaction verification is "visual cryptogram" [62]. When a user initiates a transaction on a banking site, an encrypted visual challenge corresponding to the transaction (consisting of a matrix of coloured dots) is displayed on the user's browser. The user uses a cameraphone (pre-loaded with required software and key) to take a snapshot of the visual challenge, and then the transaction details are displayed on the phone's screen. A confirmation code is generated if the user choose to approve the displayed transaction, which is then typed into the PC browser for confirming the transaction.

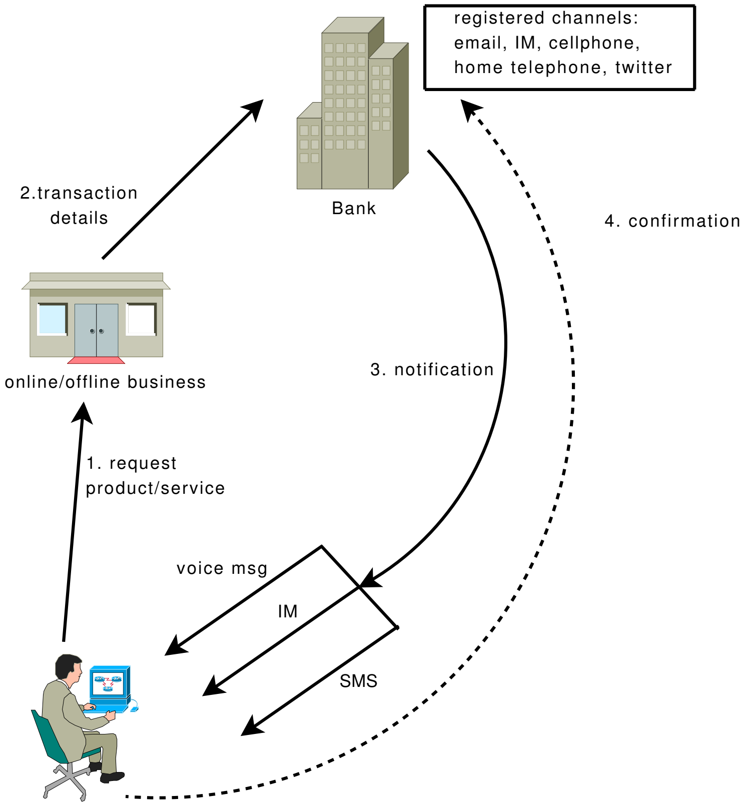


Figure 4.9: Notifying a user through user-selected channels

4.2.2 Notification through Multiple Channels

In this model users are notified by their banks when a transaction is performed on their account (debit or credit accounts). Notification channels may include email, SMS, Instant Messaging (IM), microblog services (e.g., twitter), phone call, and voice mail. Users can select one or more communication channels of their choice, and when a transaction occurs on a user's account, the corresponding bank sends details of that transaction to all user chosen media. Banks may require users to act on a notification message by confirming the transaction, or the notification message can simply serve as a *log message*. In the later case, users can check the messages at their convenience, e.g., putting all messages under a certain email 'label' or 'folder' and check them daily.[11] Alternatively, banks can summarize all transaction for a day in a single email. This instant or daily notification apparently puts users in control, and reduces

[11] Most email clients, including some web email applications facilitate rule-based classification (based on e.g., 'from' address and keywords in a subject line).

the burden of checking all transactions from a monthly statement at once. Certain transactions such as issuing of a new credential, high-value transfers should require phone/voice/video confirmation (see e.g., VideoTicket [175]). To counter automatic approval from a malware-infected personal device (e.g., cellphone, PDA), *physical presence* mechanisms (e.g., a hardware switch, vertical/horizontal shaking) of Trusted Platform Module (TPM)-enabled devices [265] may be used.

In this model, we assume that users will select one or more notification channels of their choice, and keep those (e.g., email address/twitter) up-to-date with their financial institutions. Security of this proposal lies on the following assumptions: (i) attackers cannot control all user-selected channels; (ii) attackers cannot suppress notification messages originating from a bank or notification service provider; and (iii) users will *carefully* check those messages. User interface design can positively influence how diligently users check notification messages (see e.g., [269]). Notification service providers also must communicate to users that these messages contain information only about transactions without any *actionable* items (e.g., live links/URLs or phone numbers to callback) which can lead to phishing attacks or malware infection.

A legitimate concern about this approach is information overload, especially for those users who make several transactions (through e.g., online banking, credit/debit cards) a day. However, in general we believe that receiving at least a summary message (email/IM/twitter) each day (when non-zero number of transactions are performed) may be easily accepted by users. Also we should bear in mind that an average user possibly receives several phone calls, SMS messages, or voice mails each day. Average Internet-connected users also regularly deal with an increased number of emails, IM messages, and messages at social networking (e.g., Facebook Wall) and microblogging sites (e.g., twitter). For sending notification messages directly to a user's voice mail, services such as slydial (`slydial.com`) can be used. At the end, users are in control of how many notification messages they want to receive, and through which channels.

Real world examples of notification services close to our idea already exist. Online financial management sites such as `mint.com` and `yodlee.com` notify users through email or SMS in certain cases, e.g., when suspicious transactions are detected or when an account's balance reaches a certain threshold. These sites can also aggregate all financial accounts (e.g., bank and credit card accounts) into one place for easy tracking. Users must sign-up for online banking and share their banking passwords with these services to enjoy the benefits of account aggregation. However, some banks prohibit sharing of passwords in their agreement, and from security and privacy perspective, such sharing could increase risk. We argue that sending notification messages directly from banks or credit card companies is inherently more trustworthy, and avoids the additional risk of trusting a third party.

4.3 Concluding Remarks

For protecting online transactions in the current untrusted environment, several academic proposals have been made recently. Financial institutions have also deployed several practical measures to deal with these attacks. We summarize several of these to put our proposed MP-Auth protocol into context; we also also expect this comprehensive review of current literature and existing solutions will motivate future proposals.

New attacks (e.g., Trojan.Silentbanker [255], web-rootkits [130], Kaminsky DNS attack [137]) have surfaced which can defeat existing counter-measures including two-factor authentication (e.g., [190]), and schemes using stand-alone smartcard readers (e.g., [88]). To win in this arms-race, we propose two simple non-cryptographic techniques complementary to existing cryptographic measures, including MP-Auth. These techniques assume multi-channel communication between users and service providers (such as banks), and may help users stop worrying about their financial credentials, and feel more confident about online transactions. However, the proposed techniques as introduced are preliminary; our goal is to motivate research in this direction assuming existing cryptographic techniques will eventually be defeated. Our proposals introduce additional burdens on users, and as such lead to decreased usability; it seems apparent, however, that some price must be paid for increased security, albeit less clear whether the particular tradeoffs are tolerable in practice, for some set of users.

Chapter 5

Privacy-Enhanced Sharing of Personal Content on the Web

Publishing personal content on the web is gaining increased popularity with dramatic growth in social networking websites, and availability of cheap personal domain names and hosting services. Although the Internet enables easy publishing of any content intended to be generally accessible, restricting personal content to a selected group of contacts is more difficult. In this chapter, we introduce an authentication mechanism for sharing personal content online in a privacy-friendly manner. We focus on improving sharing mechanisms of personal content, and addressing the compromise of web servers hosting such content.

5.1 Introduction

Through social networking and photo-sharing websites, and personal blogs, it is becoming increasingly popular to make personal content available on the Internet. For some users, these sites provide a textual and/or pictorial documentary of life. Primarily because it is the easiest mode of operation, many users of these services allow their personal web content to be accessed by all other Internet users, often with the false impression that none other than their family or friends would look into their personal online posts [177]. Privacy concerns are largely being ignored (sometimes unknowingly) in the current rush to online *lifecasting*.

Social networking websites such as Facebook and MySpace provide access control mechanisms for partially restricting personal content to a known circle of contacts; photo-sharing websites such as Flickr and Shutterfly provide similar mechanisms. A user can invite her friends and family to be added to her *permitted list*, and can authorize only such people to view her web content, but only if they create accounts at the publishing user's social networking site. Although users reportedly disclose personal data in abundance at these social networking sites, a relatively small number of users limit access to their profiles only to a friends' circle; several studies provide evidence of such behaviour [110, 177, 213, 284]. While this limited restriction might help users' privacy, this applies only for the content on those (few) sites.

We focus on the general problem of privacy-enabled web content sharing from any user-chosen web server. Many users now own domain names for hosting personal websites, facilitated by the very low price; as of October 2007, a top-level domain name may cost less than \$6/year, with \$4/month commercial hosting fees. Most ISPs also offer free web spaces for home users. It is thus cheap and easy to make any personal data available to anyone around the globe through a website; however, restricting such content to a selected group of people is more difficult. Currently this is achieved primarily by either (i) advertising an obscure link through personal email, i.e., a URL which is not linked from any other webpage; or (ii) protecting a webpage with a password, and distributing that password among chosen contacts through email, instant messaging (IM), or phone.

Emailing an obscure URL to many contacts (friends and family members) is a rather cumbersome approach, especially if the shared URLs are often updated. Password protection (e.g., HTTP Authentication [93], forcing a login dialogue/page) is not uncommon among the more technically inclined, but this leads to yet one more password to share and maintain, and once a password is shared with someone, the access grant cannot be retracted without changing the password (which also requires distributing the new password to all other contacts). Also, anyone who learns the shared password can view the protected content without the publishing user's consent; anyone knowing the password can pass it on to others, and such transitive access is not generally preventable.

Relying on the immense popularity of public instant messaging (IM) networks,[1] we propose a scheme called IM-based Privacy-Enhanced Content Sharing (*IMPECS*) to disseminate personal web content by leveraging the established *circle of trust* on IM networks; by a circle of trust we mean the mutually *trustworthy* relationship (to some extent) as formed through the IM contact list feature which requires explicit user permission for being added to another user's contact list. We assume both publishing

[1] For example, according to one estimation [16], there are about 350 million user accounts in MSN and Yahoo! IM networks in total.

and viewing users can, or already do use IM. A user's web content can be viewed only by her IM contacts. Further restrictions can be applied depending on which group of users (e.g., family, friends, co-workers) a specific contact is placed in by a *publishing user*, i.e., one who originally makes personal content available for her IM contacts. A *viewing user* is one who wants to view such content. We assume that a web server and an IM server share a user-specific content sharing key; a *ticket* (similar to a session cookie) is generated by the IM server for a viewing user using the content key of a publishing user, and the web server validates the ticket before serving data from a user's web folder (cf. Kerberos [184], ORiginator CONtrolled (ORCON) access control [105]).

Our primary goal is to enhance privacy (i.e., confidentiality) of users' personal web content; we do not aim for very high-end or military-grade security, as the security of IMPECS is limited by the underlying IM and web communication protocols, which in current practice transfer most content in plaintext although authentication passwords are generally sent over SSL (cf. [155, 43]). The main intended feature of IMPECS is that total strangers are precluded from direct access to a user's personal web content, but "friends" as designated by the user's IM contact list are allowed access (without requiring any special shared password). IMPECS also prevents large-scale web crawlers and auto-indexers from tagging personal data and pictures (see e.g., [11, 157]). However, malicious IM contacts of a publishing user may of course re-post the user's private content to a public web forum, and we are not proposing any form of digital rights management (DRM) control.

In summary, IMPECS offers the following features and benefits.

1. Privacy-Enhanced Sharing. A publishing user's personal web content can be viewed only by the IM contacts that she pre-approves. Thus privacy of a user's web content is restricted to a designated group. For many existing IM users, such groups can be leveraged without additional setup costs.

2. Usable Security. The privacy enhancement does not require a viewing user to separately update his IM client, or remember the publishing user's URL, or have access to a site-specific password to view the publisher's content. Similarly,

the publishing user need not carry out any extra steps beyond existing management of an IM contact list, although finer granularity lists can optionally be created by advanced users.

3. Interoperability. In contrast to social networking websites, a user can publish her web content at any web server of her choice, and yet be able to maintain greater access control on her content.
4. Decreased Risks Related to Sharing. By restricting open access to personal details, IMPECS reduces opportunities for launching context-aware, targeted phishing attacks [183, 238, 282].
5. Protection Against Web Server Compromise. A variant of IMPECS (Section 5.4) can prevent en masse *drive-by-downloads* [205, 252] as currently being enabled by the compromise of a hosting provider with a large number of customers.

To test our design, we built a prototype of IMPECS using the IETF standardized Extensible Messaging and Presence Protocol (XMPP [220, 221], i.e., the Jabber IM protocol). This required only minor modifications to the IM server, and PHP scripts on a web server. Our implementation source code is available on request.

Organization. In Section 5.2, we discuss the proposed IMPECS scheme, threat model and operational assumptions. Our prototype implementation is discussed in Section 5.3, along with brief comments on deployment issues. A variant of IMPECS is discussed in Section 5.4. Section 5.5 provides further motivation, an overview of existing and proposed work related to personal content sharing, and a comparison of IMPECS with these in terms of user convenience and usability. Section 5.6 concludes.

5.2 IM-based Privacy-Enhanced Content Sharing (IMPECS)

In this section, we describe the proposed IMPECS scheme, threat model and operational assumptions. Table 5.1 summarizes our notation. We assume readers are familiar with basic IM definitions such as *presence* and *contact list* (e.g., see [154]).

Overview of IMPECS. Assume user A maintains a website on a web server S_w. A registers her site with an IM server S_i, and sets permission for the site, e.g., which

A, B	Two IM users Alice and Bob, both members of each other's respective contact lists. A is the publishing user; B is the viewing user.
S_i, S_w	IM and web servers, respectively. Both A and B have accounts with S_i, and A maintains an account with S_w.
ID_{Aw}	A's user ID at S_w (unique in S_w's domain).
K_{Aw}	A's content sharing key, shared with both S_w and S_i.
$\{data\}_K$	Authenticated encryption [99, 28] of *data* using symmetric key K.
URL_A	The URL of A's publishing web folder on S_w.
R	Access restrictions on URL_A as imposed by A.
T_{iw}	An access control ticket for viewing URL_A (generated by S_i, and validated by S_w).
URL_{AR}	A *registration* URL generated by S_w when requested by A. The content sharing key and restrictions are shared between S_w and S_i through this URL.
URL_{AT}	A *viewing* URL (for accessing URL_A) containing a ticket T_{iw}, generated by S_i at the request of B.

Table 5.1: Notation used in IMPECS

contacts can access which pages/folders. For example, contacts in the group "friends" may have different permissions than the group "family." S_w and S_i share a user-specific content sharing key for A. IM contacts of A can see (through their IM clients) whether A offers any personal URL which they are permitted to view. When a contact B wants to visit A's advertised personal website (or any pages thereon), B sends a request to S_i to visit the website. Depending on restrictions R (e.g., duration, frequency) for viewing webpages at URL_A, S_i generates a *ticket* (similar to a session cookie), and sends a special URL to B along with the ticket. B receives the URL instantly (e.g., as an IM text message) from S_i, and can visit URL_A within a time period as specified by the ticket. Note that A need not be online to provide this permission. We now describe the scheme in greater detail.

Setup. A and B are two IM users who maintain IM accounts at the same IM server S_i. (Note that A and B may use different IM servers, as long as their IM servers facilitate communication between the users, e.g., as in distributed XMPP [220], Windows Live/Yahoo! Messenger.) Both users have added each other into their contact lists; adding someone to a contact list requires explicit permission from the user being added (a common practice in most IM networks). A also puts B into an appropriate group of her contact list (e.g., "family", "friends", "co-workers"). A maintains an account with a web server S_w, and uploads some personal pictures or files under a web folder URL$_A$ at S_w. A wants to share URL$_A$ with a select group of IM contacts including B.

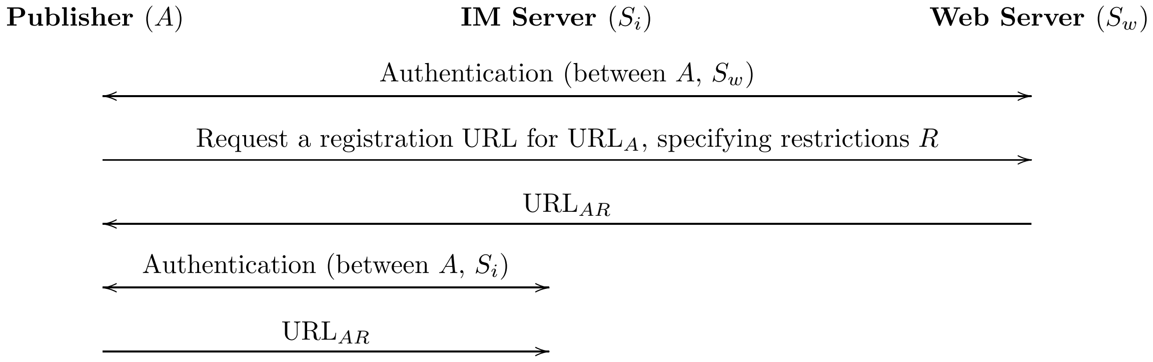

Publisher (A)
IM Server (S_i)
Web Server (S_w)
Authentication (between A, S_w)
Request a registration URL for URL$_A$, specifying restrictions R
URL$_{AR}$
Authentication (between A, S_i)
URL$_{AR}$

Figure 5.1: Registering a URL in IMPECS

Registering a URL with the IM server. We now describe the steps for publishing a content-hosting URL in IMPECS. Figure 5.1 outlines the following steps.

1. A logs into S_w (e.g., using a pre-established password over SSL).
2. A uploads her personal files and sets restrictions on URL$_A$, e.g., the length of time a ticket will remain valid after being generated by S_i (using e.g., HTML check-boxes or drop-down lists). A then requests S_w to generate a registration URL for URL$_A$.
3. S_w generates a random content sharing key K_{Aw} (e.g., 128 bits, sufficient for precluding offline dictionary attacks) and stores it in a protected database, or in a file under A's private space. S_w constructs the registration URL, $\mathrm{URL}_{AR} = \texttt{http://<URL}_A\texttt{>/?userid=}ID_{Aw}\texttt{\&key=}K_{Aw}\texttt{\&restrictions=}R$, and sends URL$_{AR}$ to A (e.g., through HTTPS). Here, by $\texttt{<URL}_A\texttt{>}$ we mean the actual URL (without the *scheme name*), not a label for that URL (i.e., not the string "$\texttt{URL}_A$").
4. A logs into S_i (e.g., using her regular IM password over SSL).
5. A forwards URL$_{AR}$ to S_i, for the purpose of registering this information with S_i. S_i stores URL$_A$, ID_{Aw}, K_{Aw} and R for future ticket generation.

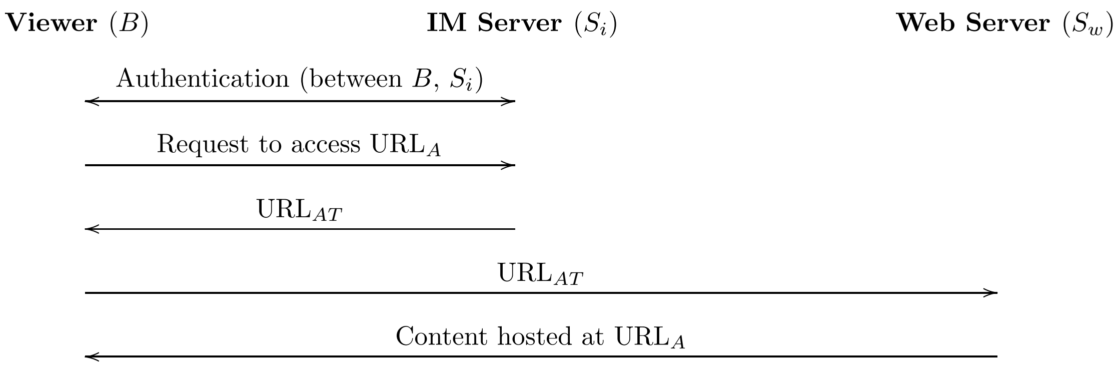

Viewer (B)
IM Server (S_i)
Web Server (S_w)
Authentication (between B, S_i)
Request to access URL_A
URL_{AT}
URL_{AT}
Content hosted at URL_A

Figure 5.2: Viewing a personal URL in IMPECS

Viewing a protected URL via an IM server. We now describe the steps for viewing a content-hosting URL in IMPECS. Figure 5.2 outlines these steps.

1. B logs into S_i (e.g., using his regular IM password over SSL), and receives his contact list as usual in IM. As part of IMPECS, B also receives a list of private URLs, offered by his contacts, which are authorized to be accessed by B.

2. B sends a request to S_i for a ticket to view one of these URLs, say A's web content at URL_A.

3. S_i generates a ticket $T_{iw} = \{ID_{Aw}, R\}_{K_{Aw}}$, constructs $\mathrm{URL}_{AT} =$ `http://<URL`$_A$`>/?userid=`ID_{Aw}`&ticket=`T_{iw}, and sends URL_{AT} to B.

4. B forwards URL_{AT} to S_w. S_w retrieves K_{Aw} using ID_{Aw} as embedded in B's request. Then S_w decrypts the ticket T_{iw}, and compares whether A's user ID in the URL is the same as inside the ticket. S_w also checks the restrictions; e.g., R could be as simple as a timestamp, in which case S_i encrypts the current timestamp into the ticket and S_w accepts that ticket if received within a specific time period (e.g., 60 seconds, as set by A).

5. S_w sends the content hosted at URL_A to B after validating B's ticket T_{iw} in URL_{AT} (as in step 4). If a valid ticket is not supplied, S_w denies access to URL_A.

Caveats. A malicious user B can compromise the privacy of content hosted at URL_A, by making a copy of the website and posting it on a publicly accessible site, or sending a valid ticket to anyone B wants. Although A cannot stop copying of her personal content, she may limit (to some extent) forwarding of a valid ticket with the help of S_i and S_w in the following way. S_i can encrypt B's current IP address into the ticket, and S_w can check whether it receives the ticket from the specified IP address as embedded inside the ticket (assuming both S_i and S_w have access to the same IP address of B).

If a content key K_{Aw} is leaked, anyone can generate valid tickets with that key, and thus compromise the privacy of content hosted at URL_A. If A changes her content key K_{Aw}, this threat can be minimized. Note that A's modifications to her web content, and key updates, are transparent to viewing users. Although valid tickets can be generated with a compromised K_{Aw}, this key does not enable access to modify A's content on S_w.

Most IM and web accounts are currently authenticated by user-chosen (generally *weak*) passwords. A compromised IM account enables an attacker to add any malicious link (as personal URLs) to that account. A compromised web account enables an attacker to post any content on the compromised user's web space, and modify content keys (although he cannot update the content key at S_i). However, these threats exist currently for both IM and web accounts; IMPECS does not increase these existing risks nor does it attempt to address them.

If user content is distributed across many different hosting sites (rather than being concentrated only to few sites as in current social networking sites), then an adversary cannot easily track users by collecting their personal web content from only a few selected sites. However, in IMPECS if the IM server S_i is compromised (or cooperates with the adversary), privacy of user content is lost for all IMPECS users of S_i even if their content is hosted at different providers; from compromised content keys, anyone can generate valid tickets for accessing user data. Thus the IM server is a potential single point of privacy breach (if compromised or hostile).

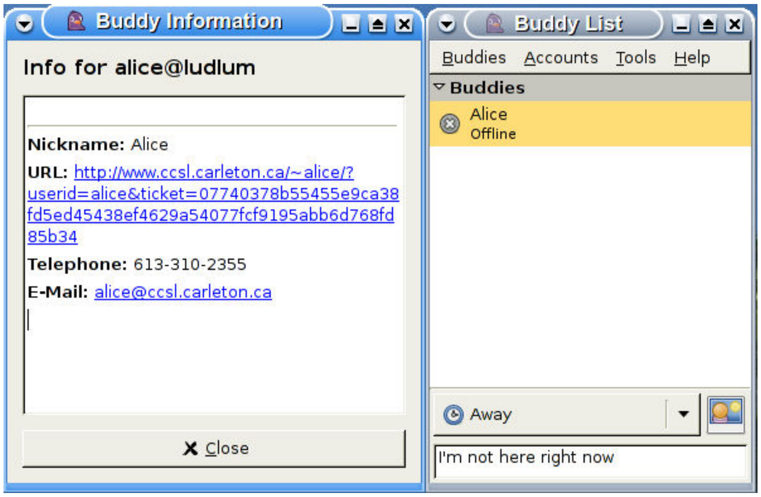
If attackers can compromise the web server of a publishing user A, they can display whatever content they want from A's site, or spread malware to users visiting the site [205]. Compromise of a web server that hosts content from a large number of users is particularly more risky, and has been reported in the past (e.g., [252]).

We briefly outline a variant of IMPECS to mitigate such a large scale compromise in Section 5.4.

Threat model and operational assumptions. We assume that the circle of trust as built into IM networks is reliable, i.e., a viewing user is not malicious. A publishing user A cannot be added to anyone's contact list without being explicitly approved by A (as is the common practice in most IM networks). To achieve fine-grained access control, we also assume that a publishing user groups contacts appropriately, and authorizes access to these groups conscientiously (e.g., which group can access which URLs). IMPECS trusts that the IM server checks publishing user A's permissions properly, and only sends tickets to authorized users. The web server is trusted to deliver A's content only after validating an appropriate access control ticket. The availability of usable site maintenance tools (e.g., HTML editing, file uploading) is also assumed for publishing users.

If a publishing user A's IM client offers a user interface for setting a personal URL (which is the norm in many IM clients, e.g., Yahoo! Messenger), we can use that to send the registration URL (containing the content key and restrictions), and thus may avoid changing A's IM client. A viewing user B's IM client can also remain the same if it offers viewing IM contacts' personal URLs (e.g., the 'View Profile' option in Yahoo! Messenger provides a 'Home Page' field in a profile webpage). We require only minor modifications to a web server through server-side scripts (assuming the server allows such scripts). The web server may optionally maintain a database of user-specific content keys; otherwise, the content key of a user must be stored in the user's private space on that web server. For an IM server, enforcing restrictions (in ticket generation) is easy; the server already restricts text (and other request) messages sent to a user from any other IM users according to the receiving user's preferences. However, users must register their URLs with the IM server; most IM services currently enable users to register personal URLs on their profiles. Leaking these URLs (without the corresponding content keys) will not by itself authorize access to any web content; they are inaccessible unless someone gets a valid ticket from the IM server.

Communication in most public IM networks (client-server and client-client) and web servers (client-server) is plaintext, although a password for authentication is generally sent over SSL. Note that our design involves the content key K_{Aw} (i.e., URL$_{AR}$) being sent over SSL. An attacker with access to the communication link may eavesdrop on private content of a user when the user uploads content to the web server, or when content is served to a (valid) viewing user. Using a variant of IMPECS (see Section 5.4), or at the added cost of SSL, these attacks can be addressed.



Info for alice@ludlum

Nickname: Alice

URL: http://www.ccsl.carleton.ca/~alice/?userid=alice&ticket=07740378b55455e9ca38fd5ed45438ef4629a54077fcf9195abb6d768fd85b34

Telephone: 613-310-2355

E-Mail: alice@ccsl.carleton.ca

Alice
Offline

Away

I'm not here right now

Figure 5.3: A viewing URL instance in IMPECS

5.3 Implementation

In this section we discuss our prototype implementation, and computational and deployment costs of IMPECS.

We implemented a prototype of IMPECS using the Extensible Messaging and Presence Protocol (XMPP [220, 221], based on the popular Jabber[2] IM protocol). As XMPP server and client, we chose `jabberd2` [128] and `Pidgin` [200] (previously known as `Gaim`) respectively, on a Linux platform. For cryptographic library, we use `OpenSSL` and the PHP `mcrypt` module; we use `AES-CBC-128` for symmetric encryption, and `/dev/urandom` for random number generation. `MySQL` is used for database support. Our implementation source code for the prototype is available on request.

[2] `www.jabber.org`

We assume that the publishing user A can run PHP scripts on the web server S_w. S_w also stores A's content sharing key in a database. We create a web folder for A on S_w which is accessible for writing (and viewing) when A logs into S_w. Other than login as A, for viewing any content of the folder, one must supply a ticket containing a valid timestamp (and ID_{Aw}) encrypted under A's content key. We assume that system clocks of S_i and S_w are (more or less) synchronized. S_w checks whether a requesting URL contains a valid ticket; we accept a timestamp to be valid if it arrives within 60 seconds of being generated by S_i. A and B also add each other to their respective contact lists.

XMPP uses the `vCard` [66] format for personal profile information storage, which facilitates advertising one's personal URL. We use this field in vCard for storing a user-specified URL, and added one field called `content-key` into the vCard table for storing a user's content sharing key (along with ID_{Aw}).[3] Ideally an XMPP user can set vCard values from any XMPP client. However, as the Pidgin implementation we used (version 2.0.1) lacks any such user interface for setting vCard values, we directly inserted URL$_A$ and K_{Aw} to A's vCard table on the jabberd2 server database. For viewing a contact's vCard, a user can select the contact from the Pidgin contact list, and choose the "Get Info" option from the context menu. When S_i receives such a request for A's vCard from B, S_i retrieves A's content key K_{Aw}, and generates a ticket by encrypting the current time and ID_{Aw} with the key. S_i then constructs a URL using URL$_A$ as the base, and ID_{Aw} and the (hexadecimal encoded) ticket as parameters. Figure 5.3 shows one example of S_i's response to B. Then B can click on the link and be able to view URL$_A$, if validated by S_w.

Computational and deployment costs. In addition to retrieving A's vCard information from a database (as required by a regular XMPP server), IMPECS requires one symmetric-key encryption by S_i. One symmetric-key decryption is required by S_w when a viewing URL is received (for ticket validation). S_w also must generate a 128-bit random number when A requests a registration URL (for the content key

[3] Instead of inserting the content-key field, ID_{Aw} and K_{Aw} could be embedded into the URL field, allowing S_i to remain in conformance with the vCard standard.

generation). These operations are relatively light-weight for the IM and web servers; no practical deployment barrier in terms of performance is expected.

In a distributed IM service such as XMPP or Windows Live/Yahoo! Messenger, where A and B may have accounts with different IM servers, IMPECS does not require any changes to B's server or client software. (Note that as of Feb. 2008, XMPP is supported by several large IM services, e.g., Google Talk, IBM Lotus Sametime, and AOL/ICQ.) We require changes to A's IM and web servers. The changes in S_w are mostly achieved through PHP scripts. A's content key and restrictions can be stored in a file under a private folder (on A's web space), or in a database if S_w provides database access. Also, B remains anonymous to S_w in IMPECS; i.e., B does not need an account at S_w for viewing A's content, as opposed to social networking websites (although a ticket is required in IMPECS). Note that all publishing users at S_w can reuse the same PHP scripts for our scheme; i.e., users are not required to write or modify the PHP scripts (these scripts may be provided by, e.g., S_w or the open-source community).

Why not to implement IMPECS as a Facebook application. For ease of deployment, we could implement IMPECS in Facebook Platform[4] or Google OpenSocial.[5] Instead we chose to base our IMPECS design and implementation on IM for the following reason. We believe that storing relationship information and user data at the same site may undermine privacy; for example, a single entity then learns too much about users and may use that knowledge to launch unfriendly (in regard to users' privacy) campaigns such as targeted advertisements, sharing user data with government agencies and third-party businesses. This also makes such sites an attractive target to compromise. These threats are quite evident from the short history of Facebook and MySpace. IM networks have also been targeted for malicious purposes such as spreading worms and phishing URLs; however, such attacks generally compromise relationship information (i.e., email addresses) but not user content.

[4] http://developers.facebook.com/

[5] http://code.google.com/apis/opensocial/

5.4 A Variant of IMPECS

In this section, we briefly outline a variant of IMPECS that can prevent malware-spread from a compromised web hosting provider. We have not implemented this variant yet.

Some large hosting providers (e.g., `godaddy.com`) currently facilitate web hosting for thousands of personal and corporate sites. If many IMPECS users host their content at such a provider, a successful attack against the provider might possibly affect all those IMPECS users. The compromised user sites could be used for malicious purposes, e.g., hosting malware for *drive-by-downloads* [205, 252]. This could be particularly bad for IMPECS users as private URLs as shared through IMPECS may appear to be more trustworthy. Here we outline a proposal that can guard against such en masse exploits.

Additional steps during URL registration. The following additional steps are required from a publishing user.

1. A uses a local application (in-browser JavaScript plug-in or an independent content editing application) to generate an encryption key K_{enc}, 128 bits long. A then uses K_{enc} to encrypt her personal files and upload the encrypted result (i.e., $\{data\ files\}_{K_{enc}}$) to S_w. This is done at the beginning of step 2 in URL registration of IMPECS (see Fig. 5.1 in Section 5.2).
2. A appends K_{enc} to the registration URL received from S_w before sending the URL to the IM server S_i. This is done at the end of step 3 in URL registration of IMPECS (see Fig. 5.1 in Section 5.2).

Additional steps for a viewing user. The following additional steps (although transparent) are required from a viewing user.

1. When S_i generates URL$_{AT}$ (step 3 in Fig. 5.2; see Section 5.2), it also appends the URL with K_{enc} as a URL fragment, i.e., <URL$_{AT}$>#K_{enc}. When B visits this URL, URL$_{AT}$ is forwarded to S_w but not the fragment, i.e., S_w does not receive K_{enc} (cf. [5]).

2. In step 5 (Fig. 5.2) S_w sends the requested (encrypted) content. B's browser uses K_{enc} as received from S_i to display the decrypted content.

The encryption key K_{enc} is not accessible to S_w at any time. Thus by compromising S_w, an attacker cannot control what is served to the visiting IMPECS users. Note, however, that regular visitors to such a site are not protected by this technique. The publishing user A may update K_{enc} in a similar way to the content key K_{Aw}. However, an update to K_{enc} does not mandate updating K_{Aw} or vice-versa, and both key updates are transparent to viewing users.

5.5 Motivation, Related Work and Comparison to IMPECS

In this section we discuss existing and proposed work related to personal web publishing, and contrast IMPECS with these in terms of privacy and user convenience.

Popular IM networks, e.g., Yahoo!, AOL, and Windows Live enable users to maintain a profile accessible as a webpage. Microsoft offers free web spaces for sharing personal web content (e.g., profile, photos, blogs, guestbook) through its Windows Live Spaces social networking website at `www.spaces.live.com`. Live Spaces is integrated with the Windows Live Messenger IM client. User A can control who may view her Live Spaces' webpage. A can invite friends to join the Windows Live Messenger network to view her content. A may authorize only her IM contacts (or a subset of the contacts) to view her space. Alternatively, A may make her space accessible to anyone on the web. If A's space is restricted to IM contacts, a contact B (from A's contact list) can login to Live Spaces using B's Windows Live Messenger login credential for viewing A's space. If logged into the IM network, B can also select A's profile from a context menu from the Live Messenger client; from A's profile, B can access A's space without further authentication. However, in either case, similar to the common social networking practice (e.g., as in Facebook or MySpace), B must join A's network to view any access-restricted content. In contrast, when using IMPECS, B does not need to know where his (IMPECS-enabled) IM contacts host their content.

To partially relieve users from the necessity of creating multiple web credentials, Microsoft permits third-party businesses to use its Windows Live ID Web Authentication[6] (previously known as Microsoft Passport). Similarly, Yahoo! offers the Browser-Based Authentication[7] (BBAuth) service that enables third-party web applications to be authenticated through widely used Yahoo! IDs. OpenID (`openid.net`) is an initiative from the open source community to unify online authentication, also reducing the burden of creating multiple web credentials. AOL has enabled the use of OpenID (through `openid.aol.com`) for its IM service and AOL Pages social network. OpenID can also be used for Yahoo! login (through `openid.yahoo.com`). Liberty Alliance (`projectliberty.org`) is another *holistic* approach to establish an open standard for online identity. If any such unified identification framework becomes widely accepted in the long-run, IMPECS would become even more appealing (e.g., through a common login credential). However, IMPECS does not address user authentication across websites per se, but rather focuses on how the existing trust network and interactiveness of a popular service like IM can be leveraged to offer privacy-enhanced personal content sharing on the web.

Most IM networks offer file sharing from user machines generally through custom-built file transfer protocols. An IM user can restrict which contacts in her IM contact list can access the shared files. However, IM file transfer protocols may not work in some cases (e.g., due to firewall restrictions), and a publishing user must be online to make her files available to others.

YouServ [23] is an end-user P2P application to enable people to easily share personal content (e.g., photos, music, presentations, work documents) with little to no cost.[8] Instead of a specialized P2P protocol, all YouServ content is served through standard web protocols (i.e., DNS with HTTP). An implementation of YouServ was used by thousands of users internally at IBM and Carnegie Mellon University (apparently the web interface for this service at `YouServ.com` is now defunct). YouServ requires two centralized components called YouServ Coordinator (for authentication

[6] `http://msdn2.microsoft.com/en-us/library/bb676633.aspx`

[7] `http://developer.yahoo.com/auth/`

[8] Note that when this research [23] was published in 2002, the cost of hosting a personal website at a third-party hosting company was much higher than today.

and peer coordination) and YouServ Dynamic DNS (for finding a peer site's dynamic IP address). A user's YouServ content remains available even when the user's PC is offline (through a peer hosted site), or firewalled (through a proxy site). Authentication is provided using a single sign-on password scheme (valid for any YouServ site). Access to any specific file can be limited to certain members of the YouServ community. Using YouServ, Bayardo et al. [22] proposed a technique to make IM file transfer easier by making local files available through transient web links; the web link of a file is sent to the recipient simply as an IM text message. In contrast to YouServ, publishing users in IMPECS make their personal content available from a third-party hosting site (as is the current common practice) instead of their own PC (or any of their peers' PC).

The popularity of social networking websites, e.g., Facebook, MySpace, Twitter, Bebo, is apparently comparable to the early years of large-scale IM networks. By joining Facebook or MySpace, users can search and connect with friends, share personal content such as photos, videos, blogs, contact information, and preferences. In Facebook, users generally locate friends from groups, e.g., classmates from the same school or university, co-workers, geographical locations. MySpace generally categorizes user groups by interests, e.g., music, photography. To add to the interactive power of IM, MySpace offers its own IM client called MySpaceIM (accessible only to MySpace users). Facebook also has recently (Oct. 2007) added IM capability through the FriendVox browser-based IM client. Twitter enables users to send short messages to selected friends through the web, SMS messages, or IM. Most social networking sites enable limited access control through explicitly creating a "friends' list." Online photo sharing website Flickr offers creation of a list of friends through Yahoo! login credentials. Other photo-sharing websites such as Shutterfly offer similar privacy-enhancing mechanisms. We discuss the effectiveness of such access control mechanisms below.

Privacy issues in social networking websites. Although social networking sites enable publishing users to partially restrict access to their personal content, privacy concerns are emerging quickly regarding the use of these networks. People have been denied or lost jobs because of their comments on MySpace or Facebook profiles (e.g., [185, 186]), a grocery chain dismissed employees for comments on Facebook (e.g., [97]), and students were suspended for their Facebook comments (e.g., [48]). Government agencies such as the CIA are suspected of tracking users with special interests (e.g., [202]); apparently under the U.S. Patriot Act, state agencies can look into a job interviewee's Facebook profile, even if the profile is "privacy-protected," i.e., permitted to be viewed only by the publisher's circle of friends (e.g., [174]). If a user removes content from his/her profile that may be deemed offensive or was posted as a momentary emotional response, or even if the user deletes the entire profile, personal content may still reside in (incremental) archives for a long time (cf. [157]).

Many users of social networking sites keep their profiles and friends list publicly accessible. A user survey [177] of social networking websites reported that 74% of adult users of those sites exposed their personal information such as email address, name, birthday, home and work address, and even Social Security Number (SSN). Only 39% of respondents chose to restrict their personal profiles only to friends. Initial results from another survey [253] of Facebook users reported that 67% of the participants kept their personal profile open for all. Another study [209] of the LinkedIn social networking website (used mostly for business purposes, e.g., to find potential clients, service providers, business opportunities, job listings) reported that people generally expose detailed and (possibly) confidential information on their profiles. Dwyer et al. [73] compared information disclosure and perceptions of trust and privacy in an online survey of Facebook and MySpace users. Facebook users were reported to reveal more identifying information than MySpace users. For example, real name, email address, and IM screen name have been disclosed by 100%, 94%, and 71% of Facebook users respectively (in contrast to 66.7%, 40%, and 49.8% of MySpace users respectively).

Gross and Acquisti [110] investigated patterns of personal information revelation and associated privacy implications using more than 4,000 publicly available Carnegie Mellon University (CMU) users' Facebook profiles. Most users provided (seemingly highly accurate) personal information including profile image, full birth date, hometown, current residence, and phone number. Personal preferences, interests, and political views were also disclosed by the majority of CMU users. Although Facebook offers privacy control, most users did not change the default privacy preferences which grant access to a user's full profile by any member of the user's groups/networks (e.g., place, institution, interest); only three CMU users' profiles (0.06%) were precluded from view by *unconnected* users (i.e., not a friend or friend-of-a-friend). Based on the revealed personal information, the authors outlined a number of privacy implications including online and real-world stalking, digital dossier of participants (by any third-party), and demographics and face re-identification (i.e., relating seemingly anonymous data to explicitly identifying information). The authors also discussed how a user's SSN may be estimated from disclosed birth date, hometown, current residence and phone number. A similar study [86] on 20,000 MySpace user-profiles reported that 68% of users kept their personal profiles open for all. Almost half of a randomly selected 1000 users' group provided global access to all elements of their personal profile. Rosenblum [213] analyzed privacy risks of social networking sites, including privacy options as provided by major networking sites and limitations of such privacy settings. In addition to highlighting privacy issues of social networking sites, Barnes [21] emphasizes that a significant educational effort from parents, schools, social networking sites, and government agencies, is required to address the emerging privacy issues related to these sites.

Jagatic et al. [132] collected publicly available "circles of friends" data from several social networking websites by using web crawlers; this enabled the researchers to quickly build a database of tens of thousands of relationships. When a (benign) phishing attack was launched by using the collected social network database, 72% of social networking targets fell victim to the phishing attack, while only 16% of regular users were fooled by the attack. In fact, social networking websites are specifically being targeted for launching *context-aware* phishing attacks (see e.g., [183, 238, 12, 282]), spreading spyware [45] and malware [239], and even for building botnets [226]. Cross-site scripting flaws in the MySpace website have been reported [262] in the past which could have been exploited to disclose even privacy-protected user content. Social networking websites with personal details of millions of users would also seem to be lucrative targets to online attackers (e.g., for targeted phishing or identity theft), and government agencies (e.g., for tracking citizens' digital identities). Equifax, a leading consumer credit reporting firm, has recently (July, 2007) warned [211] that user profiles on social networking sites are a "goldmine" for ID thieves. MySpace acknowledged [2] that as of July 2007, it had removed more than 29,000 registered sex offenders profiles from the MySpace website, indicating that criminals with other than monetary motives are also exploiting the abundance of personal information freely available at social networking sites.

Ahern et al. [6] examined privacy decisions in mobile and online photo sharing using Flickr. Most interviewed users in the study showed little or no concern regarding exposure of aggregated contextual information, e.g., time, location (embedded with some uploaded photo files), arising from their photo-sharing habits. In addition to manual photo-tagging as offered by common photo-sharing websites such as Flickr and Shutterfly, Polar Rose (`www.polarrose.com`) uses facial recognition algorithms for tagging unknown images of a subject if there is a tagged image of the subject on Polar Rose's image database (see [11] regarding the inadequacy of current privacy laws in this regard). Search engines, e.g., Spock (`www.spock.com`), customized for finding personal profiles posted at different websites, may provide even easier access to personal web content. Since September 2007, Facebook is allowing non-members to search for user profiles that are not access-restricted; third-party search engines such as Google and Yahoo! are also authorized to index such profiles (as of Feb. 2008).

Convenience and usability of IMPECS. IM contact lists are already in place for IM users, whereas social networking sites require users to invite friends and family members through, e.g., email to join a user's "friends' list"; sometimes these standardized, impersonal invitation emails simply irritate the recipients. IM is more interactive than social networking sites despite the immense recent popularity of those sites. For many IM users, IM clients start automatically after users log into their PC, and many IM users remain signed-on to an IM network as long as they use their computer. Social networking sites require a user to open a web browser, load a site, and sign into that site for maintenance or to view a friend's profile. IM users can view and control more effectively what content is being shared at any given time;

information regarding who viewed what, and how frequently, may also be gathered from the IM server's ticket-issuing statistics.

We believe the following factors make IMPECS appealing. The viewing user B's role in IMPECS is simplified in comparison to the current social networking practice. B need only log into his IM client, and select an intended contact's URL for viewing. In contrast to social networking sites, B can remain unaware of who hosts his contact's web content. B need not even store or memorize A's URL; in fact, a bookmarked URL may not work depending on A's restrictions. However, B must realize that private URLs as shared through IMPECS are different than regular static URLs. The publishing user A's content sharing key K_{Aw} must be shared between S_i and S_w. This can be accomplished by any of the following means (in increasing order of convenience): (i) A manually copies the registration URL (containing K_{Aw}) from S_w to S_i using an interface provided by her IM client; (ii) S_w forms an XMPP URI (`xmpp:` [222]) embedding the key with URL$_A$, and A activates the URI (e.g., by a mouse click) to be processed by a locally installed XMPP client;[9] the client sends URL$_A$ and K_{Aw} to S_i; or, (iii) S_w forwards K_{Aw} to S_i if there exists a pre-established relationship between the servers. A content key update is also similar to updating a URL link at S_i. To revoke B's viewing permission, A can simply place B on a separate IM contact group which does not have access to URL$_A$ (or remove B from her contact list). Thus it is natural to expect that IMPECS is more convenient than current content sharing/limiting techniques on the web (e.g., password protection, obscure links). However, we hesitate to make any stronger usability claims without formal user testing.

Once published on the Internet, private content may become permanent, e.g., through archived search engine queries and web crawlers [157]; in essence, the Internet does not forget anything published on it, although much of the personal information on the web (e.g., blogs, emotional responses, criticisms of friends and authorities) is meant to be transient. Unfortunately, momentary emotional responses to an event, if posted as text or image on the publicly accessible Internet, may bring unpleasant consequences at a later time. Our approach can enhance *forgetfulness*

[9] Most popular IM protocols provide custom URI handlers, e.g., `ymsgr:` (Yahoo! Messenger), `aim:` (AOL Instant Messenger).

of the web by not making personal content public in the first place (cf. [43]). Web pages meant for certain personal contacts, friends and family will remain among the pre-established circle of trust as long as none of the trusted IM contacts make copies of a web page and republish it on the public Internet.

5.6 Concluding Remarks

Privacy is typically violated as a consequence of any of a number of factors. These seem to include: (i) oppressive administrations or large corporations (sometimes by exploiting the common misconception of "I've got nothing to hide" [248]); (ii) a shortage of usable tools to guard online privacy; (iii) apathy towards privacy; and (iv) a misunderstanding of the implications of lost privacy. In our opinion, easy access to usable privacy tools may change the actions of ordinary web users towards online privacy; IMPECS is designed to be such a tool to enhance privacy of personal web content (i.e., we focus on addressing factor (ii) as listed above). We leverage the existing circles of trust among IM contacts, as well as encourage further refinements of trust in popular IM networks. Unlike current social networking websites, users do not need to (re-)build a "friends' list" in parallel to IM contact lists. In addition, users can publish their content at any website of their choice, and still be able to maintain privacy of their content (without being limited to use only a particular social networking site). Note that the general idea behind IMPECS extends beyond IM and IM circles of trust; any equivalent scheme, (ideally) containing pre-arranged groups, could similarly be leveraged (cf. Liberty Alliance People Service [151]).

As reported in a user survey [177], even most adult users of social networking websites keep their personal profiles open for all. We believe that such behaviour results largely from practical issues such as difficulties in ensuring close contacts join the same social networking site as the publishing user (just to view a friend's profile), or simply ignorance of the privacy implications of posting personal details on the Internet. IM is a very popular Internet application, possibly with a greater user base than social networking sites. Distributed IM services such as XMPP and Windows Live/Yahoo! networks enable IM communication between users of different IM networks. Therefore, we believe that IMPECS has significant deployment advantages over other personal content sharing techniques (e.g., password protection). By restricting personal content to a closed group of IM contacts, we believe IMPECS reduces opportunities for launching context-aware, targeted phishing attacks [183, 238, 282] where fraudsters collect social context of a target victim from their seemingly innocuous unprotected personal data, and enhances *forgetfulness* [157] of transient personal content on the web.

Chapter 6

Digital Objects as Passwords

Our proposed protocols such as MP-Auth (Chapter 3) and IMPECS (Chapter 5) involve the use of user chosen text passwords. Such passwords may be viewed as the *weakest link* to security in those protocols (and in many other existing password-based protocols). To improve the strength (i.e., entropy) of these passwords, in this chapter we introduce a password generation technique that relies on digital content controlled by or accessible to everyday users.

6.1 Introduction and Motivation

Despite all their shortcomings, text-based passwords are still heavily used by everyday users and security experts. Decades apart independent studies reveal that people consistently choose *weak* passwords [170, 90, 233]. There are several apparent reasons for such behaviour. Strong or high-entropy passwords are difficult for users to generate, memorize, and reproduce at a later point in time. Also as the benefits of a strong password over a weak one are not readily noticed, there is little apparent motivation for users to spend extra effort in choosing strong passwords. Blaming users, or restricting password choice with complex rules (see e.g., [163, 246]) usually do not help. Alternatives to text-based passwords such as biometrics, hardware tokens, and two-factor methods are still far from being widely deployed, and password use is likely to dominate user authentication in the foreseeable future [94]. The existence and indeed recent rise in SSH password-guessing attacks[1] indicates that stronger passwords still increase security despite the proliferation of password stealing attacks, phishing and keylogging (cf. [91]).

[1] One experimental setup [208] reported an average of 2,805 SSH login attempts per computer per day.

Let us assume that with continual education, motivating efforts [92, 276], restrictions or proactive checking [292], users are persuaded to choose strong (random-looking, high entropy) passwords for everyday use. These strong passwords generally remain usable (i.e., well-remembered) only if used often. However, passwords employed to access rarely-used services, or in *secondary authentication* (e.g., when a user has lost/forgot the primary password) are not frequently recalled, motivating users to choose weak passwords/secrets that are difficult to forget or obvious when given a hint. We introduce an Object-based Password scheme called *ObPwd* which may be best used for passwords that are (i) infrequently used, or (ii) used for secondary or fall-back authentication, e.g., Password Verification Questions (PVQs); see for example, Rabkin [206] for a discussion of serious security weaknesses of PVQs as used in a number of current online banking sites.

The basic idea of ObPwd is the following. Many users currently possess a large collection of digital content such as photos, mp3s, and videos. Much of this content is *mobile*: users may keep it on personal devices (e.g., USB sticks, cellphones), or upload it to personal sites (in some cases, password-protected). Many users also have instant access to static content from the web, e.g., Internet Archive (`www.archive.org`), Project Gutenberg (`www.gutenberg.org`), and Google Books (`books.google.com`). An ObPwd password can be generated from such digital content as follows: compute a hash of user selected content, such as a photo file from the user's USB stick, and then convert the hashed bitstring to a password (a *random-looking* string of keyboard characters or as an option, a human readable sequence of words using existing techniques [115, 161, 114, 214]). Users keep a record (memorized or written) of a pointer to their content used in generating each password. Users can write down the password in a secure place, or re-create it from the content when needed. ObPwd requires no modifications to the software interface of password-based systems. Also, authenticating parties (remote or local) are not required to be aware of ObPwd (e.g., storing of a user's password-generating objects is not required).

ObPwd may offer the following benefits over existing techniques (see also Section 6.3).

1. Reduced Memory Load. Instead of requiring users to remember exact passwords or passphrases, ObPwd only expects them to recall a semantic *pointer* to

their password object (e.g., hints for an image, video, entire/partial document, executable, URL, or highlighted text passage from a webpage).

2. Resistance to Offline Dictionary Attack. Without having access to all of a user's possible password objects (from local media and web), attackers cannot build a password dictionary. Assuming password objects significantly vary among users (e.g., each user may have an independent collection of photos), creating a generalized password dictionary for ObPwd seems impractical. In contrast, building a dictionary of popular passphrases is apparently feasible [145], and general password dictionaries are already available [192].

3. Written Record of Passwords. In contrast to most graphical passwords [254], users can easily keep a written copy of ObPwd passwords (e.g., in a safe place as backup). Thus ObPwd enables converting an image-based password into human readable text (which also facilitates sharing – see below), and benefits from the easy memorability of object or image hints while keeping the simplicity of text passwords (easy deployment, written records).

4. Password Sharing. In cases where objects are already shared (e.g., photos, documents), ObPwd allows safer password sharing through a hint or description of the password object, without transmitting the actual password over the network. This seems preferable to some current practices such as sending shared passwords over email. It also allows sharing of the text-form output, which although often discouraged, may nonetheless be an important usability feature.

6.2 Object-based Password (ObPwd)

In this section, we discuss the ObPwd scheme in more detail, threat model, variants of the basic idea, and a prototype implementation.

Threat model, operational assumptions, and notation. We assume that password-generating objects in ObPwd are selected from a large public collection, e.g., files (including pdf) from the ACM digital archive (containing millions of archived documents), or from personal digital content (inaccessible to others). Hopefully either the large size of pools of such source objects, or the inaccessibility of private content will impede attempts to build offline dictionaries. Ideally users would not choose password objects from their (publicly accessible) personal website or public profiles as in Facebook/MySpace sites. (However, appending such objects with a salt apparently reduces some risks; see under 'Variants' below.) To enable *access-from-anywhere*, users either carry password-generating objects with them, or have online access to those objects. ObPwd passwords, and hints (text reminders) to password objects can optionally be written down. Passwords must be written down if a user does not want to carry content files with her.

If a password is directly generated from the password object and users *copy-paste* that password instead of typing it in (see 'Implementation' below), keylogging attacks on passwords may be restricted. However, if ObPwd is used in regular web login, we strongly suggest that the password objects should be stored in local media (i.e., user devices) when passwords are generated on-the-fly (right before login). If a password is re-created from (plaintext) web content the following attack is possible. An attacker observes or records traffic from the intermediate network looking for a user to go into a content-hosting site right after or before requesting an authenticating website; thus the attacker can capture or narrow down candidates for the password-generating content. When ObPwd is used for encryption/decryption in a user's local media, getting access to password-generating objects from the network does not allow the attacker to gain any protected content (as the network attacker does not have access to the user's local encrypted files). Of course if the attacker already controls the user PC, neither ObPwd nor other password schemes can help. Similarly, this scheme is vulnerable to shoulder surfing and phishing (but see 'Variants' below). When a user has multiple password objects for different accounts/applications, the usual issue of password interference may also surface (which object is used for which account). However, ObPwd is focused to increase usability of a *strong* password by leveraging distinctive object choices that might be made by a user, including leveraging their personal content. Table 6.1 summarizes our notation.

Steps in ObPwd. The steps in ObPwd are as follows; see also Figure 6.1.

1. U selects an easy to remember object M from her personal media or from the web. To preclude offline dictionary attacks and predictable object prefixes,

U	An ObPwd user.
M	A password object selected by U for a particular site/application.
$h(\cdot)$	An appropriate cryptographic hash function.
Hash2Text$(\cdot)$	A function (e.g., based on [115, 161, 114, 214]) for converting hashed bits into a string of keyboard characters, or optionally, words.
pwd	A password as generated by Hash2Text$(\cdot)$.

Table 6.1: Notation used in ObPwd

M should be required to exceed a minimum size (perhaps 30 bytes). Considering the time that may be required to hash very large objects (in step 2), such as a movie, M is ideally truncated to an appropriate number n of bytes (e.g., $n = 100000$).

2. U indicates the selected object to the ObPwd tool, which generates the hash H of M using a secure hash function h: $H = h(M)$.

3. H is used to generate $pwd =$ Hash2Text(H).

   H may be truncated depending on the required size of an output password. pwd (and M) should not be stored at the same place or media as the protected content. If used as a site password, pwd may require special encoding depending on the particular site; we do not address encoding issues separately here (but note that encoding techniques are addressed elsewhere [214]).

Variants. The basic idea of ObPwd can be extended as follows. A user-selected, ideally memorable *salt* string (s) may be appended as a second input to the hash function h: $H = h(M, s)$. The salt could be a 4-digit PIN, or a dictionary word. This enhancement may impede attackers even when a user's password object is exposed, albeit at the cost of memorizing a salt string. If used only rarely, then the salt need not be memorized but rather could be looked up from where it was written down. While the ObPwd scheme as proposed is vulnerable to phishing attacks, this weakness can be addressed, by appending the URL of a target site (as in [214], this can be done without user involvement) with the password object, i.e., $H = h(M, url)$.

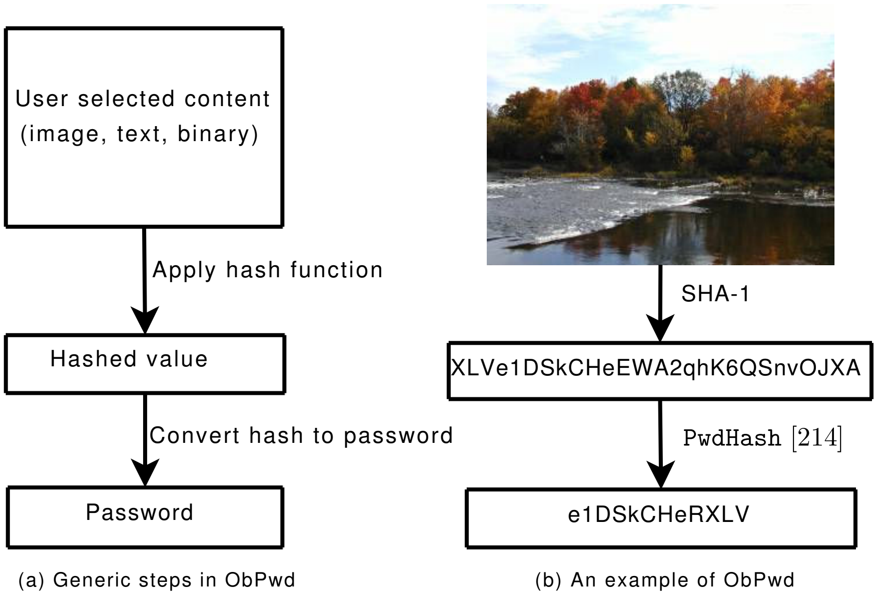


Figure 6.1: ObPwd steps with an example

Implementation. We have implemented a basic prototype of ObPwd as a browser extension for Firefox (Fig. 6.2), and also as a stand-alone application in Windows XP (developed in `C#`). When a user clicks the right mouse button on a web object (an image, highlighted text, or a file URL), the browser extension inserts a menu item (e.g., "Get ObPwd from Image" in Fig. 6.2) into the context menu; if selected, the extension generates a password from the underlying content and displays the password in a dialog box (Fig .6.3). In the local application, a user selects a particular file, which is then used as the password-generating object, and the password is displayed in a text box. For both implementations, we use SHA-1 as the hash function, and PwdHash [214] for converting hash values into a password (12 characters long, alphanumeric). We use at most $n = 100000$ bytes from a password object, and require a minimum of 30 bytes. Both implementations are available online (see Section 6.4). For mobility, if the ObPwd extension or application is not available from a remote computer, a website for generating passwords from user objects could be designed (cf. `pwdhash.com` [214]). We do not make any claims about the usability of the present prototype, but if the idea generates interest, would hope to pursue this and to host such a site.

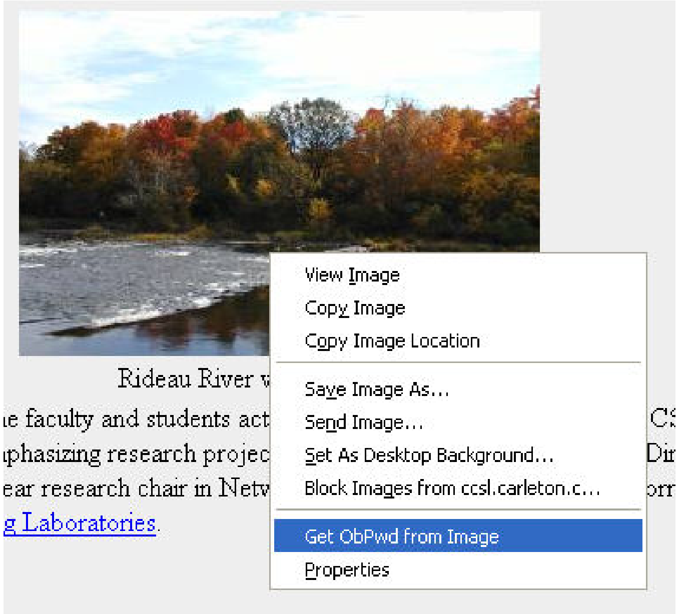
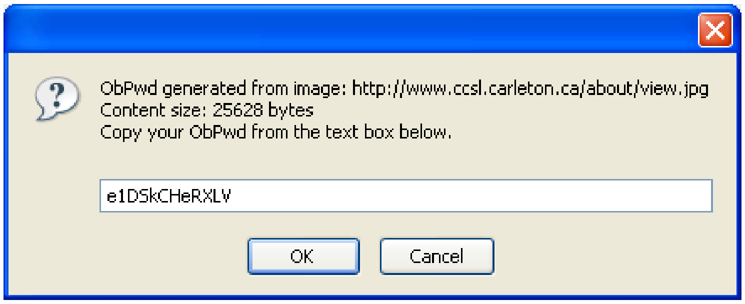

Rideau River w
ie faculty and students act
iphasizing research projec
ear research chair in Netw
g Laboratories.
CS
Dir
orr
View Image
Copy Image
Copy Image Location
Save Image As...
Send Image...
Set As Desktop Background...
Block Images from ccsl.carleton.c...
Get ObPwd from Image
Properties

Figure 6.2: ObPwd extension menu in Firefox



ObPwd generated from image: http://www.ccsl.carleton.ca/about/view.jpg
Content size: 25628 bytes
Copy your ObPwd from the text box below.

e1DSkCHeRXLV

Figure 6.3: Password generated from the selected image

6.3 Related Work and Comparison

There have been countless publications on passwords. Here we discuss only a selective subset of schemes designed to strengthen passwords (i.e., improving entropy) or to enhance usability (i.e., improving the ease-of-use). Infrequently used passwords such as Personal Verification Questions (PVQs) are discussed separately as ObPwd is apparently most suitable for these.

6.3.1 Schemes for Improving Password Strength/Usability

Cheswick [50] proposed an obfuscated challenge-response based authentication scheme assuming people can *compute* a simple response to a given challenge according to a (user-selected) *pass-algorithm*. Both the challenge and response are obfuscated with *decoy* information. This scheme offers several desirable features (e.g., protection against keyloggers and phishing). Challenges noted by the author include users may forget the pass-algorithm/obfuscation technique more readily than a regular password, if used infrequently.

Florêncio et al. [91] argue that relatively weak passwords (e.g., with 20 bits of entropy) may provide enough security for web accounts assuming that: (i) a *three-strike* type rule (i.e., login is blocked after three failed attempts) is deployed to counter brute-force attacks; (ii) the user ID space is much larger than the IDs in actual use; and (iii) the valid user ID list is not readily available to attackers. Meeting these assumptions requires assistance from authenticating sites.

To improve password strength while maintaining usability, Forget et al. [92] proposed Persuasive Text Passwords (PTP) wherein system-generated characters are inserted at random positions into a user-chosen initial password. Users can accept the proposed password, or request (until satisfied) alternate suggestions. PTP essentially provides a middle ground between system-chosen (strong but difficult to remember) and user-chosen (weak but memorable) password schemes.

Yan et al. [291] conducted a user-study to compare regular user-chosen passwords, random passwords and mnemonic phrases. They reported finding that mnemonic phrases are as good as random passwords, and easier to remember. However, passphrases (and mnemonic passwords generated from them) may also be attacked by building a dictionary from commonly used phrases as available on the web [145].

Disk encryption software TrueCrypt allows users to use any local file along with a possibly empty password as an encryption key.[2] Users cannot write down the actual encryption key as a backup, and the generated key is used only with TrueCrypt. ObPwd was conceived independently.

[2] This feature is apparently available since version 4.0 (Nov. 2005); see `http://www.truecrypt.org/docs/?s=keyfiles`.

Apparent advantages of ObPwd. In addition to web authentication, ObPwd passwords can arguably be used for applications which must withstand offline dictionary attacks (e.g., file encryption). Also, the deployment of ObPwd does not require any changes in system-side processing or password verification, or to the user interface in a web or local application.

ObPwd enables converting image-based passwords into text, and thus may be viewed as a middle ground between text and image-based password schemes. ObPwd can use (memorable) images while retaining simple advantages of text passwords (no-cost deployment, written records). While some people think writing passwords down and sharing passwords are poor practice, this arguably depends on the threat model, and usage. Certainly, being able to write down and backup infrequently used passwords seems essential. The fear of not writing down passwords may also encourage users to choose weak passwords.

Sharing of passwords (quite common in the real world; see e.g., [196]) in most graphical schemes is awkward if not impossible. ObPwd may enable better password sharing than text and graphical schemes without sacrificing confidentiality to third parties. For example, if two users share a digital photo folder (e.g., through personal media), then one user can choose a specific image as the password object, and send the other user a hint or description of the image (e.g., "our whitewater kayaking photo") over public media or email. Now an eavesdropper can see the hint,[3] but cannot generate the shared password without having access to the image object itself. Although this is in effect equivalent to sharing a list of secret keys, arguably the advantage here is that we use more *meaningful* objects than randomly generated keys.

6.3.2 Personal Verification Questions

Personal Verification Questions (PVQs) are used for resetting a forgotten password or as part of login. While generally weaker than passwords, PVQ answers are typically equally useful to access an account. The availability of personal information on the web has apparently made it easier to correctly guess PVQ answers [206] (see also [109, 232]). As an example, Hollywood actress Paris Hilton's private photos and close

[3] Here we assume that the hint is not an obvious link to a publicly-accessible web object.

contacts' phone numbers were exposed when an attacker was able to log into her T-mobile web account by answering her pet dog's well-known name to a PVQ [152].

Academic work on PVQ. Early work on PVQs includes *cognitive passwords*[4] and a related user study by Zviran and Haga [300]. It was reported that users could recall cognitive passwords more accurately than regular passwords. The authors also tested guessing attacks on cognitive passwords by significant-others of a user. Fact-based questions such as "mother's maiden name" and "name of your best friend in high school" were correctly guessed by 57% and 43% of users respectively. Opinion-based questions such as "favourite colour" and "last name of your favourite college instructor" were correctly guessed 41% and 10% of the time, respectively. The authors used 20 questions, of which users must answer five randomly selected questions at each login attempt.

Ellison et al. [76] proposed using personal questions and answers for recovering secret keys. Instead of using a passphrase, they require a user to pre-register n personal questions and answers (usually low-entropy), and then recover the secret key by correctly answering some $t < n$ of the questions. Thus a user can forget some answers, but still recover the secret.

Recently, Rabkin [206] analyzed over 200 PVQs as used in 17 financial websites. Taking the "era of Facebook" into account, different classes of attacks are considered (e.g., random guessing, attacks automatically using online information, dedicated human attackers, and knowledge through personal acquaintance). As a possible defense, the following use of personal content was suggested. A user may upload an image of a person, and an answer to the question "what is the name of this individual?" However, as noted, any tagged photo of that person enables attackers to answer the question.

ObPwd as PVQs. If ObPwd is used in certain types of PVQ schemes (which allow free-format questions/answers), attackers cannot succeed without getting the actual password object. Many PVQ answers have quite low entropy ("What is your favourite colour?"). ObPwd password entropy is expected to be significantly higher.

[4] These are questions and answers related to a user's personal facts or opinions; they were designed to be used as regular passwords.

Also ObPwd requires no uploading of multimedia content to an authenticating site, and an exact copy of the password object is required for a successful attack (cf. [206]).

6.4 Concluding Remarks

Humans are not good at choosing high-entropy secrets that are easily memorable for a long time. Arguably, current password generation techniques and password-restricting rules have largely failed to improve password strength. Creating passwords from personally meaningful/memorable digital objects may be more user-friendly than any existing password rules; we emphasize, however, that we have not yet carried out any user testing. Depending on the application, variants of the basic ObPwd scheme may be suitable; for example, URLs can be appended to password objects (as in PwdHash [214]) if phishing is a concern.

Apparently passwords generated by our method would have more entropy than regular passwords. We have yet to devote serious attention to the question of determining defendable estimates of the security gains that might result, or a method to quantify guessability in the absence of very large-scale user trials (e.g., of millions of users). Indeed, despite existing password crack papers (e.g., [288]), it is not clear that the community even has a strong understanding of the empirical security of existing text passwords chosen by the mythical *typical user* for the mythical *typical password application*. Studies of even as many as 500,000 users are too small for the long-tailed distribution of user-chosen passwords, and obtaining or publishing cleartext passwords in such studies is complicated by privacy concerns [90].

Our proposal has obvious limitations. Losing the pointer or the password object itself (if no written copy is kept) is equivalent to forgetting a regular password. Also, obvious and publicly accessible choices of password objects, e.g., the profile photo of a user's Facebook account, could result in even less security than text passwords. The potential security of ObPwd relies on the richness of the universe from which public objects are selected, and/or the inaccessibility of personal objects. ObPwd is introduced here to solicit feedback and promote discussion, to help advance the eternal quest for a better password scheme. We encourage readers to try out our implementation available at `http://www.ccsl.carleton.ca/~mmannan/obpwd/`.

Chapter 7

Localization of Identity Numbers for Addressing Data Breaches

In this chapter and the following (Chapter 8), we explore a design philosophy called "design for damage control," which focuses on restricting threats from compromised sensitive data assuming that we cannot *perfectly* secure such data. Here we introduce a novel *localization* technique to reduce the value of personal identity numbers in case they are breached from (often large) corporate/government databases.

7.1 Introduction and Motivation

Currently personal identity information is stored in a number of different places including small and large corporations, government agencies, educational institutes, hospitals, and financial data processing centers. Coupled with such data replication, insider abuse (e.g., [49]), negligence (e.g., [112, 55]), inadequacy of existing technology for protecting user data, and a computing environment arguably "under occupation" [146] (by e.g., malicious software and semantics attacks) have resulted in numerous large-scale data breaches. The U.S. Government Accountability Office (GAO) defines data breach as "an organization's unauthorized or unintentional exposure, disclosure, or loss of sensitive personal information, which can include personally identifiable information such as Social Security numbers (SSN) or financial information such as credit card numbers" [102]. Data breaches from organizations, small and large, considered to be highly secure or otherwise, make the news almost every day, and now seem to be the business norm. Beyond simple credit card numbers, leaked information now includes SSN, drivers' licenses, dates-of-birth, and bank account numbers. Aside from privacy exposure, these breaches facilitate *identity* *fraud*,[1] heavily exploited by underground criminal networks. For example, according to Symantec [256, p.23], an individual's *full identity* (which may include name, address, date of birth, SSN, driver's license number) can be bought for only $1-15. One primary reason for the enormous demand of compromised personal records is that most existing ID numbers are static, and thus reusable elsewhere (especially where the corresponding physical ID token is not required or the token can be easily forged).

In response to large-scale data breaches, security proponents have placed increased importance on data encryption, use of sophisticated intrusion detection technologies, etc. However, these conventional techniques are still not widely deployed, and also have been subject to a continual stream of innovative attacks, from *side-channel analysis* of cryptographic keys (e.g., timing/power analysis attacks), to the recent *cold boot attacks* [113]. Additionally, such technologies are of limited help in the case of organizational mismanagement.[2] In the financial sector, as credit card number disclosures increased, some banks started to offer one-time use credit card numbers for online transactions around Sept. 2000. Several research proposals (e.g., [218, 149, 169]) have been made focusing mainly on enhancing such credit card number generation and user-friendliness.

We argue that improving data security mechanisms or new legislation for protecting consumer information is of limited use. Identity fraud originating from data breaches will grow as more and more identity information is collected and stored digitally.[3] In an environment where data breaches are evidently inevitable, it is our main thesis that the use of static/reusable ID numbers should be reduced, if not completely eliminated, to fight identity fraud. Building on existing ideas and experience regarding disposable credit card numbers, we propose a more general approach and technique to deploy an *ID number localization* approach to restrict and/or detect abuse of a wide variety of sensitive personal identification numbers. Our use of the term *localization* is primarily intended to mean customizing ID numbers to a specific

[1] We define identity fraud as unauthorized exploitation of credential information through the use of false identity [176].

[2] According to one study [273], 87% of the breach cases analyzed could have been prevented with "reasonable security controls."

[3] "...the difference between such crimes [ID theft] today and in the future is the scale of the data involved" [285].

relying party, which need not be tied to a particular geographic or physical location. We also outline four variants of our main proposal. Despite these proposals, our primary goal is to increase awareness of the new environmental attack paradigm by which ID numbers ultimately become compromised; different solutions are explored here to motivate further research in this area.

An individual may be required to provide their SSN or driver's license number to several parties (employers, banks, credit reporting and car rental agencies), all of whom store sensitive identification details for a long time. Confidentiality of such data may be breached by any of these parties. If a person has worked for five different companies in the past, her SSN may leak from any of those, and once disclosed may facilitate identity fraud. If a localized SSN scheme were in place, where each employer would get a *different (non-reusable) version* of the SSN of the given individual, then a disclosure of any such SSN would not be useful for identity fraud. We base example solutions on this idea and explore several variants. Again, our fundamental assumption is that user data will eventually be breached (cf. [263, 146, 47]) primarily through relying parties; we focus on how to nonetheless mitigate identity fraud.

In a broader sense, one obvious reason for the severity of current identity fraud, spam, phishing, and many other Internet-related attacks is the leverage gained by using data compromised from one site at many others, repeated times. This *compromise once, reuse multiple times* feature provides significant advantages to attackers. Our approach is a defensive paradigm of (virtual) localization for the *use* of credential information on the Internet and in the physical world. Localized identification numbers as generated by our scheme are valid only for a particular relying party. This apparently reduces the value of compromised credential information to attackers, thereby reducing the threat and also the cost to defend adequately. Our approach attempts to undermine the asymmetric leverage attackers currently enjoy.

In summary, our contributions and discussion points include:

1. New Paradigm for Protecting Personal Identity Information. Legislative and technical efforts such as encryption alone to better secure personal identification data are evidently inadequate in today's untrusted computing environment. As one response, we propose the use of localized, restricted-use

identification numbers instead of static, reusable ID numbers to limit large-scale identity fraud.

2. Breadth of Scope. We focus on protecting all types of identification numbers in general instead of solely credit card numbers. Where most previous solutions focus on card-not-present transactions, we address both card-present (ID-present) and card-not-present transactions. Furthermore, our approach addresses breaches resulting from real-world (offline) incidents such as lost or stolen disk drives, and backup tapes (i.e., independent of computers being compromised by malware).

3. Variations. To take into account deployment feasibility, and cost-benefit trade-offs, we explore several variants of our proposal (appropriate for varying scenarios).

Overview. We outline our main proposal along with threat model, notation and operational assumptions in Section 7.2. Four variants of the main proposal are introduced in Section 7.3. In Section 7.4 we briefly discuss related work and representative examples of recent data breach incidents. Section 7.5 concludes.

7.2 ID Number Localization

In this section, we outline our proposed ID localization scheme. Threat model, notation and operational assumptions are also discussed here.

Overview. A credential issuing party provides each user a smart card (chip-card) with a unique identification number for the user and a *secret* key both stored on the chip and in print on the card itself.[4] For example, the credential issuing party for SSNs is the Social Security Administration (SSA), a user's SSN is a unique ID number (issued by SSA), and a secret key is a random string of digits or characters of sufficient length (e.g., 128 bits). As the secret key is stored on an ID card, the user does not have to memorize it. Using a software program (preferably on a trustworthy platform) or a chip card reader, a user generates a (virtual) localized identification

[4] The printed *secret* key is used when a chip card or card reader is unavailable (variant 2 in Section 7.3); see item 1 under "Assumptions."

number for a credential relying party from the issued identification number, the secret key, and the *registered* identifier (e.g., a business name) of the credential relying party. For verification, the relying party forwards its registered identifier, the localized SSN, and the user's name and address to the issuing party. From name and address, the issuing party can uniquely index or identify the user, re-create a localized SSN, and verify whether the supplied SSN is valid (i.e., was created with the *right* key). In essence, the proposal turns fixed (long-term) ID numbers into secrets that can be verified, but not reused across relying parties.

Threat model and notation. ID numbers are compromised in many different ways, including data breaches (through e.g., compromised merchants' databases or data-centers, and lost/stolen disks and backup tapes), phishing attacks, *dumpster diving*, corrupt insiders, workplace, theft of purses, wallets, or postal mails, social engineering, and existing/past relationship with victims. If a user's physical card is stolen or lost, valid localized numbers may be generated and used unless the user promptly reports the incident to card issuing parties (or the card is protected otherwise, e.g., through a traditional PIN). We focus on preventing identity fraud from large-scale data breaches, instead of attacks that are not much scalable (i.e., difficult to carry out in a comprehensive fashion). We primarily consider breaches of personal ID numbers that can be directly used to perpetrate identity fraud; breaches of other sensitive information, e.g., records of a person's health and education, business secrets (which are also commonly exposed), although important, are out of our scope. We assume that ID number issuing parties can be relied on to protect their customer credentials. User data is breached mostly from relying parties as ID numbers issued by one entity is generally used (and thus replicated) by many relying parties. Such replications increase the possibility of a breach. The following notation is used:

I, U, R	Issuer, user, and relying party respectively.
U_F	User's long-term fixed ID number (issued by I).
U_R	User's localized ID number for R.
K_{IU}	Long-term secret key shared between I and U.
$f_{K_{IU}}(\cdot)$	A cryptographically secure MAC function f, keyed by K_{IU}.[5]
U_A	Lookup data (e.g., name and address) of U.

[5] To be more precise, $f(\cdot)$ should be a Pseudo-Random Function (PRF), as similarly used in "independent OTP" [217] and `PwdHash` [214].

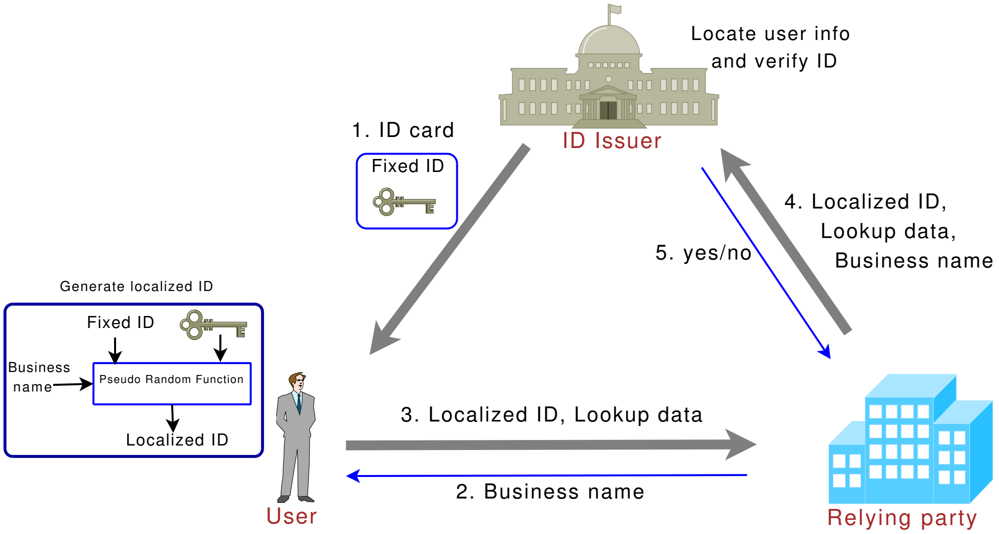


Figure 7.1: ID number localization scheme

Detailed steps. The steps required for ID number localization are as follows (see also Fig. 7.1).

1. The credential issuer (I) provides a smart card to U with an ID number U_F (unique in I's domain), and a secret key K_{IU} upon verifying U's identity (e.g., through an in-person visit or equivalent). U_F is directly used only with I, and only I and U know U_F and K_{IU}. Additionally, I also keeps U's lookup data (e.g., name and address) U_A associated with U_F and K_{IU}.

2. In order for U to generate a localized ID number for the relying party R, R sends information facilitating the localization (e.g., R's business name) to U.

3. In response to R's request, U generates a localized ID U_R for R.

$$U_R = f_{K_{IU}}(U_F, R) \tag{7.1}$$

   U sends U_R and U_A to R. The MAC output may require modifications to conform with the target ID format. For an on-site (card-present) transaction, U_R is generated using U's chip-card at R's chip-card reader (e.g., simply by *swiping* the card). The reader provides the relying party's name[6] to the card

[6] Additional relevant information may be provided as well; see item 4 under "Assumptions".

for computing U_R; U does not input anything explicitly. For card-not-present (e.g., web) transactions, U may input the relying party's name to her chip-card reader. (See variant 2 in Section 7.3 for localized ID generation without a chip-card or chip-card reader.)

4. To verify the validity of U_R (i.e., whether U_R has been generated from K_{IU} and U_F), R sends (U_R, U_A, R) to I.

5. Using U_A, I locates U_F and K_{IU}, and checks the validity of U_R; i.e., from U_F, K_{IU} and R, I generates U_R as in equation (7.1) and compares it with the received U_R. I then sends the verification result (accept or reject) to R. Appropriate integrity must of course be provided in this latter communication.

Assumptions. Operational assumptions in our main proposal and its variants (see Section 7.3) are as follows.

1. We assume a user does not reveal the printed long-term key on the card to any third party (e.g., through phishing attacks). If chip-cards are used and users generate ID numbers only using an available chip-card reader, printing the secret key on the card can be avoided.

2. For a variant of our proposal (variant 2 in Section 7.3), we use a user's personal device (cellphone or PC). Such a device may expose the long-term user key if it contains malware. However, we focus on large scale data breaches, rather than individual information leaks (through malware, phishing, or shoulder-surfing).

3. In our main proposal, users must keep their lookup data (e.g., name and address) U_A updated with an ID issuing party. For variants 1, 3, and 4 (Section 7.3), this assumption may be relaxed. Arguably it is impractical to expect users to notify all their ID issuing parties of address changes. However, *secondary* issuing parties may update U_A from *primary* parties that are generally expected to have the most recent U_A information, e.g., banks, credit bureaus.

4. We assume that a localized ID number as generated in equation (7.1) is tied to a particular relying party, and can be reused at the same relying party but not

anywhere else. This assumption allows *traceability*,[7] and apparently increases usability by requiring less user input (cf. [169]). However, our generalized proposal can be extended to generate more restricted ID numbers (even transaction-specific numbers), e.g., by including timestamp, validity period, transaction amount, etc. along with the name of a relying party (R) in equation (7.1). Such an extension may restrict insider abuse, and reuse of compromised IDs even at the same relying party from where the breach occurred.

5. In our localized ID scheme, an issuing party is directly able to keep track of the usage of a customer's ID. However, information aggregation by a centralized entity (e.g., credit reporting agencies, personal background check for law enforcement) from multiple sources is no longer straightforward under our proposal (due to unlinkability among different custom ID numbers). To achieve such aggregation, we assume that ID issuing parties will collaborate when required/appropriate, for example, if compelled by law enforcement authorities. Note that for variants 1, 3, and 4 (Section 7.3), aggregation remains unaffected.

In contrast to many current uses of identity information, in our proposal, verifying that identity information is *valid* involves the relying party carrying out a communication with the issuing party. This is part of the price we pay for the added security.

7.3 Variants

Here we discuss four variants of our main proposal. These variants are outlined to initiate further discussion, and for now we defer an in-depth analysis of implementation details, deployment strategy and associated costs, though critically important for rolling out any of these variants.

Variant 1: Localized authorization code. The localized ID scheme above uses U_R in place of U_F. This requires certain formatting of the MAC output. For example, in a regular credit card number, the first six digits identify the issuing bank and the last digit is the Luhn check digit. If U_R is used as a credit card number, it must

[7] Note however that, as per the current agreement between U.S. retailers and credit card companies (e.g., Visa and MasterCard), a merchant's identity may not be revealed even when the merchant is responsible for a data breach (see e.g., [172]).

conform to these restrictions (which may complicate U_R generation, depending on the ID number space of U_F). However, as such a number is identical to a real credit card number, it can be used in the existing infrastructure. An alternative approach is as follows: require the use of U_F along with U_R for a transaction, i.e., now U_R is used as a dynamic *authorization* code (cf. Card Verification Value 2 (CVV2) codes for credit cards [169]) accompanying the fixed ID number. By policy, U_F must not be accepted without a valid U_R. Now U_R need not conform to any strict formats. Existing implementations must still be changed to accommodate the extra authorization code check, but changes to many existing implementations would likely be significantly reduced, for example, databases which are indexed by U_F; this allows straightforward information aggregation from multiple sources. Theft of an U_F (or even an U_F, U_R pair) is no longer a concern, as for generating a new U_R attackers also require the key K_{IU}.

Variant 2: Without chip-card or card-reader. Some credential issuers may not adopt chip cards in the near future. Some ID cards do not even contain a magnetic stripe for storing extra or sensitive information. For example, Canadian Social Insurance Number (SIN) cards and (older) health cards contain only a user's name and ID number in a printed form. Our approach can be used in such cases if users are issued long-term secret keys (perhaps printed on the ID card itself). One-time ID numbers can be generated from a user's fixed ID number and the shared secret, using a personal computing device (e.g., a PC or cellphone containing an appropriate application). If such numbers are generated only infrequently, usability (e.g., having access to a computing device, providing user input) may not be affected much.

For frequently used ID numbers such as credit card numbers, we assume the availability of chip cards with on-site card readers. For card-not-present transactions (e.g., e-commerce), it would be easier for a user if she has access to a chip-card reader (e.g., the user can avoid typing in the secret key). However, users can still use a personal device (with appropriate software on it) for generating localized ID numbers.

Variant 3: Database poisoning. Organizations storing a large number of personal records can create legitimate looking fake records, and insert those into their databases. An issuing party may share a unique secret key with each relying party, and the relying party creates fake records using the shared secret such that the fake records are indistinguishable to an attacker (without knowing the secret key), but the verification party can detect those as fabricated and linked to a particular relying party. The proportion of fake records can be configured to the sensitivity of stored information, storage/computation overhead, and/or company policy. For example, if a compromised database contains 1% fake records, on average, the breach is detected within 100 transaction attempts (i.e., the use of compromised records).

If this technique is implemented by all relying parties of a particular ID number, it will enable the ID issuer to distinguish the compromised relying party from a specific fake record during verification. However, the issuer must assign each fake ID number such that the number is attached to a specific relying party, and cannot be generated by a relying party without the assistance from the issuing party. Also, the issuer must require (through e.g., policy) that all relying parties insert fake records consistently. Satisfying these assumptions in practice could be difficult considering current compliance failures (e.g., [60]). However, this technique is apparently easy for ID issuing parties to implement (i.e., only requires self-compliance). Relying parties may also benefit from database poisoning by reducing their long-term liabilities due to breaches; according to one analysis [273], most organizations currently remain unaware of a compromise for months (63% of cases) and even years (2% of cases).

Responses to a fake record detection may vary depending on cost-benefit trade-offs, e.g., heightened scrutiny of incoming requests, activating additional verification processes, or deactivating the legitimate ID number temporarily or permanently (blocking new uses). To its advantage, this variant does not require any assistance from users (i.e., usability cost is non-existent), at the cost of increased backend overheads.

Similar deceptive techniques have been in use for protecting postal mail addresses for a long time [268]. Inserting *honeytokens* [249] (bogus digital records) has been discussed for monitoring unauthorized access/use of different types of digital resources including databases with sensitive personal information. In the security literature, Kursawe and Katzenbeisser [146] discussed similar deceptive techniques to detect compromised personal records (e.g., a credit card number) from a user PC. For example, a user may store one valid credit card number along with 100 other legitimate-looking (but fake) card numbers. When these numbers are compromised, such an incident may be promptly detected by monitoring for the use of fake numbers. To reduce the value of the collected information by spyware in a PC, `SpySaver` [228] creates several fake web users on the PC and generates web browsing actions emulating real users with counterfeit information (e.g., email addresses, and credentials for web accounts). `HoneyIM` [290] uses *decoy* Instant Messaging (IM) contacts for detecting IM worms in an enterprise environment. When a worm attempts to spread by sending its copy to every contact of a compromised account, the worm infected PC can be easily tracked by monitoring the decoy account.

Variant 4: User-centric authorization. In this variant, we propose to actively engage users in blocking critical misuses of breached data records, e.g., issuing of new credentials, transfer/sharing of existing credentials from one party to another, and high-value transactions. Assume that a user registers her personal device with each ID issuing party. When a relying party attempts to verify the user's ID with the corresponding issuer, the issuer notifies the user's personal device (through, e.g., phone call, SMS, email). The issuer may depend on the response from the user device to respond to the relying party, or simply keep the user device informed (i.e., in terms of *log* messages). To counter automatic approval from a malware-infected device, *physical presence* mechanisms (e.g., a hardware switch, vertical/horizontal shaking) of Trusted Platform Module (TPM)-enabled devices [265] may be used.

Several techniques involving personal devices have been proposed in the recent past (possibly due to the increased proliferation of mobile phones, blackberries, PDAs). In contrast to `CROO` [176], this variant requires only critical transactions involving ID numbers to be verified through the personal device (not every transactions). Unlike "owner-controlled information" [95], the user is not expected to store and maintain all her privacy-sensitive information. The use of a personal device has also been proposed [271] as a "heartbeat locator" (securely detecting the location of the device) to counter identity fraud. A verification center continuously tracks the location of a registered device, and compares the location information with that of an attempted transaction before approving the transaction. A transaction may fail if the locations are not matched. This technique merely requires a user to keep her personal device with (or around) her, and the user does not need to interact with the device for approving a transaction. Issues related to device theft and cloning have also been discussed [271].

Note that our main proposal and variant 2 prevent ID number reuse across relying parties although a compromised ID may remain valid within the breached party's domain. Variant 1 is a prevention mechanism while variant 3 is detection-only. Variant 4 can prevent misuse if explicit user authorization is always required; otherwise, it becomes detection-only.

7.4 Related Work and Data Breach Incidents

Here we briefly discuss a few high-profile data breach incidents, and academic proposals related to our work.

7.4.1 Examples and Costs of Data Breaches

Examples of breaches are easily cited. Personal records of all 25 million child benefit recipients in the U.K., including their dates of birth, bank accounts, and national insurance numbers had been lost from a government agency when the agency mailed the records in discs [112]. Sensitive personal information on 26.5 million U.S. veterans had been reportedly stolen [55]. While the TJX data breach [58] is still fresh (affecting about 45 million users), millions of user records were stolen from the `Monster.com` job site [26]. A database admin reportedly [49] stole and sold 8.4 million customer records containing bank account and credit card information; another employee at the same company previously compromised 2.3 million records [237]. The theft of a computer with thousands of "top-secret" mobile phone numbers, information regarding undercover terrorism and organized crime investigations was reported by a U.K. company [120]. A list of prominent data breaches in the U.S. from Jan. 10, 2005 to Apr. 17, 2009 reports [203] the exposure of more than 253 million records containing sensitive personal information.[8]

Erickson and Howard [78] analyzed news accounts of data breaches from 1980 to 2006, and identified *organizational mismanagement* as one prime reason for these

[8] `Attrition.org` and Identity Theft Resource Center [125] also maintain similar but independent lists of data breaches. Verizon [273] provides a comprehensive analysis (including breach sources, attack types and paths, time span of breach events) of 500 such data breach cases from 2004 to 2007; see also [116].

breaches. Considering the incidents from 2005 and 2006, when most U.S. states legislated mandatory reporting, they found that in 68% of news stories concerning data theft, the theft could be attributed to organizational behaviour (e.g., administrative error, insider abuse). Apparently, even if we could *remove* malicious outsiders (e.g., organized crime) as an element in data breaches (e.g., through security technology), data records with sensitive personal information will still be breached in large numbers.

A Ponemon Institute benchmark study [201] investigates the costs of a data breach using data from 35 U.S. organizations for the year 2007. On average, it costs an organization $197 per record compromised, an increase of 8% since 2006 (for financial services firms, the cost is $239 per record). The cost of lost business due to a breach (from the loss of existing customers, and diminished new customers) is estimated on average $128 per record (a 30% increase from 2006). Acquisti et al. [3] provides a comprehensive analysis (using data from 1999 to 2006) of the impacts of a privacy breach incident on a company's stock market value; these effects are generally negative and statistically significant in the short term, but not so visible in the long run. Costs to consumers affected by a data breach is even more difficult to estimate. According to one estimation,[9] in 2007, the average fraud amount per ID fraud victim in the U.S. is $5720 (about 9% decrease from 2006). However, in most cases, it is difficult to clearly establish a link between breached data and fraud [102]. This fact is also often exploited by breaching parties to understate their legal responsibilities.

General observations. Large-scale data breaches occur frequently, and current legal and technical measures are failing to slow down this trend. Costs of these breaches are significant for both consumers and corporations. Establishing a concrete link between data breaches and identity fraud is often difficult because misuse may occur long after a breach, and misused information cannot directly be attributed to a particular breach incident. However, the growing underground (criminal) market for stolen personal information strongly suggests that breached ID numbers can be easily sold and exploited [256].

[9] Javelin Strategy and Research Survey (Feb. 2007); for excerpts see `http://www.privacyrights.org/ar/idtheftsurveys.htm`.

7.4.2 Related Work and Comparison

In NSPW 2007, Beaumont-Gay et al. [27] proposed a policy-based solution called Data Tethers where enforced policies are dependent on the operating environment; i.e., access control policies for stored data on a computing device differ depending on whether the device is inside a *secure* environment or otherwise. Data Tethers may encrypt or remove sensitive data when an insecure environment (e.g., stolen laptop) is detected. This technique assumes the existence of an "actually secure" computing environment, and that Data Tethers policies will always be flawlessly enforced. While such techniques can substantially improve data security in certain environments, in general, we believe the most prudent assumption is that data will be compromised, irrespective of protection mechanisms deployed to prevent such leaks.[10] Also, in many cases it is only realistic to acknowledge that the prevention of compromise is beyond the control of end-users, and of relying parties who hold such information.

Gates and Slonim [95] introduced the owner-controlled information paradigm to address the issues of "privacy, consistency and mobility" in regard to personal information. Users are expected to maintain all of their personal information, identification information, as well as medical history and financial information using a personal device. Organizations must contact a user directly to collect and use personal information. Although this technique provides greater control over a user's sensitive data, it apparently comes with several unique challenges and high usability costs (some of which have been discussed in the paper, e.g., lost/stolen device, unauthorized access, backup and recovery). Ashley et al. [17] propose a framework to addresses "privacy management" (e.g., publishing concrete privacy promises, user consent management, privacy enforcement, auditing) for collected customer data in an enterprise environment. This framework may provide higher level of privacy assurance, although it may incur significant costs (in addition to requiring an enterprise to develop a comprehensive privacy policy, and to enforce that policy honestly and consistently).

To reduce customers' fear of using credit cards online (i.e., for card-not-present transactions), several banks enable users to generate limited-use (e.g., one-time) card

[10] For example, several breach incidents have been reported from the U.S Department of Defense [104], presumably one of the most security-aware organizations.

numbers through their websites. These dynamically generated card numbers are tied to a user's fixed credit card, and can be used for online purchases instead of the fixed card itself. Examples from real-world deployments of such schemes include American Express' Private Payments,[11] Discover card's Secure Online Account Numbers,[12] and SecureClick [241].

Rubin and Wright [218] proposed an *offline* scheme for generating limited-use credit card numbers (i.e., without requiring direct interaction between a user and card issuer for generating new numbers). A user (U) and a card issuer (I) share a long-term secret key (K). U possesses a computing device, and stores K on it. To generate a new credit card number number, U selects a monetary restriction (e.g., $100 limit), expense category (e.g., food), limited validity period, merchant name, timestamp etc. and encrypts these restrictions using K (in an arbitrary finite domain encryption scheme [36]). U then transmits the newly generated limited-use number and her identifying information (e.g., name and address) to the card issuer via the merchant. From the identifying information, I selects K and verifies the limited-use number (e.g., checks the restrictions).

Assuming the availability of chip-cards and chip-card readers, Li and Zhang [149] (see also [150]) proposed a one-time credit card scheme with limited involvement of a user (i.e., no transaction specific user inputs) for card-present (on-site) and card-not-present (web, and phone/fax/email) payment scenarios. A user generates one-time use numbers simply by inserting her credit card into a chip-card reader. In this scheme, a credit card stores a secret value (S) and an initial one-time credit card transaction number (CCT). Assuming T_{cur} is the current CCT number, the next CCT T_{new} is generated by hashing (T_{cur}, S). At the end of the current transaction, T_{new} replaces T_{cur} on the card.

Using a personal device such as a cellphone, Molloy et al. [169] proposed an offline scheme for generating virtual credit card numbers similar to Rubin and Wright [218]. Instead of using finite domain encryption, Molloy et al. used a MAC to avoid several limitations of such encryption, e.g., encoding merchant names in a compact format.

[11] Introduced in Oct. 2000, discontinued since Oct. 2004.

[12] http://www2.discovercard.com/deskshop. Orbiscom's (`orbiscom.com`) Controlled Payment Numbers technology enables Discover and several other one-time disposable credit card providers.

Also, a user-memorable password may be used as the long-term shared key (P) between a user and card issuer. The MAC key is generated from the hashed output of P and the user's real credit card number as assigned by the issuer. A *transaction string* (including U_A, expiration date, R, transaction amount) is MACed to generate the virtual credit card number. The authors claim the *forgery resistant* property, i.e., an attacker cannot (easily) forge credit card numbers even if he knows the user's real credit card number and some virtual credit card transactions. However, this property relies on the assumption that the long-term shared key P is *strong* (i.e., has high entropy), which in practice may not hold for most user-chosen passwords.

The main appeal of using disposable credit card numbers is apparently to alleviate the inconvenience of customers contacting their bank (and replacing a compromised card), as users are typically liable for at most $50 in case of fraudulent use of their credit card. Generally, credit card numbers alone cannot be used for identity fraud. On the other hand, efforts to reduce misuse of more sensitive information such as SSN are apparently scarce (see e.g., limiting the use of SSNs as an identifier [75], and the FTC workshop [85]; see also [103]).

Similar to our proposal, `CROO` [176] attempts to address the generic identity fraud problem albeit by a different use of one-time passwords; for example, CROO is more complex and seeks to secure individual transactions, whereas we focus on securing ID numbers.

From a legal perspective, Solove [247] identifies several inadequacies in the traditional model for addressing privacy violations using ID theft as an example. ID theft is a "consequence of an architecture" [247] exploited by ID thieves (e.g., the use of SSNs for indexing a large array of sensitive personal information held by government agencies and private businesses). A new architecture has been proposed based on the Fair Information Practices (originating from a 1973 report by the U.S. Department of Housing, Education, and Welfare). The Fair Information Practices focus on increasing an individual's involvement (e.g., participation in the collection, storage and use) in personal information systems. As an example mechanism, Solove [247] proposes that user-chosen passwords or account numbers be used for accessing credit reports instead of using SSNs or other sensitive personal information.

Partly driven by increasing public demand, most U.S. states (44 out of 50, as of Sept. 2008) have legislated data breach notification laws, requiring organizations to report breach incidents to a state agency. While the question of whether these laws will reduce data theft in the long run is yet to be answered, it has been reported that so far their effect appears to be statistically insignificant [212]. However, another study [223] reported that notification laws are increasing "awareness of the importance of information security" among organizations surveyed. Payton [197] provides a review of current U.S. state and federal laws regarding data breaches, and possible legal remedies available to fraud victims. Costs and benefits of a national data breach notification requirement have also been analyzed [102].

Some businesses attempt to prevent identity theft by providing a service which places *fraud alerts* on a customer's credit bureau profiles. However, in one incident [283], the identity of the CEO of such a company was exploited to obtain a $500 loan (using the CEO's SSN which is displayed publicly on the company website and TV commercials).

Advantages of ID localization. Advantages of our proposal relative to existing ones include the following.

1. A localized ID number as generated in equation (7.1) is bound uniquely to the relying party. While this does not offer the advanced restrictions of Rubin and Wright [218], their additional restrictions require additional user input; thus we expect that our simpler proposal may enjoy better usability.

2. While Li and Zhang [149] assume the availability of user-level chip-card readers, our proposal (variant 2) can work when the user has access to a wide variety of computing devices (e.g., cellphone, PC).

3. Localized ID numbers are computationally immune to offline dictionary attacks as they do not rely on user-chosen passwords (in contrast to Molloy et al. [169]).

4. ID localization may also limit *synthetic ID theft* [123] where imposters use real identifiers (e.g., SSN) along with other fake attributes, e.g., name, address.

7.5 Concluding Remarks

Once personal ID numbers are collected by third parties, we believe that the most prudent assumption in today's Internet environment is that they will be breached at some point in time, despite best efforts (if any) of the collecting parties. In addition to lost personal privacy (e.g., medical history, purchase habits, online and real-world activities under surveillance), these breaches enable large-scale identity fraud. Some of these fraudulent activities remain undetected by their victims for years [84, 273]. While direct monetary losses for consumers from such fraud are recoverable to some extent, nonmonetary damages (productivity/time lost to resolve identity theft [257], denied credit or other financial services, harassment by debt collection agencies, criminal investigation or arrest [24]) are not; see e.g., the FTC 2003 report [84]. One of the main problems is that agencies/corporations responsible for these breaches of customer records are not generally held accountable for the breaches, and presently there is no significant financial penalty. We expect that if it was corporate data that was being compromised, corporations would pursue legal remedies; but since it is primarily the personal information of individuals, and the perceived dollar amount likely to be gained through legal remedy is small compared to the cost of litigation, individuals generally do not pursue legal remedies.

We outline an ID number localization approach and its variants to reduce identity fraud due to large-scale data breaches that expose reusable fixed ID numbers. Barriers to deployment include existing databases indexed by ID numbers (such as SSN), and legacy information aggregation applications. However, localized IDs may reduce liability of businesses when a breach occurs. Our proposals may also ease the burden of following security *best practices* or governmental regulations for protecting consumers' identity data. These techniques may also enable out-sourcing customer data to countries with different rules and regulations, or enforcement reality.[13] However, in the end, natural adoption of our proposals may not occur in the absence of imposing increased liability on relying parties for breached data, strong consumer lobbying, and perhaps government legislation/regulation.

[13] An undercover journalist was reportedly [261] able to buy U.K. customers' bank accounts, credit card details, passport and driving licence information from call center workers in India.

Rather than focusing on analysis of a particular solution, our proposed variants here are intended to initiate further discussion on how to better address the current problem of identity fraud resulting from breached databases of personal information on millions of customers. There are certainly deployment challenges with several of our proposals; consequently, we raise the question, “Are there better proposals that can address the same problem?” We believe that fundamentally new approaches are required to address this problem, which clearly is not addressed by existing solutions.

Chapter 8

Salting/Localization of PINs for Addressing Flawed Financial APIs

In this chapter, we expand our “design for damage control” viewpoint (see Chapter 7), and apply it to restrict exploitations of compromised banking PINs. Several *PIN cracking attacks* have been reported in the recent past which may reveal user PINs by abusing design flaws in widely deployed financial APIs. Despite best efforts from security API designers, flaws are often found, and even APIs with a formal proof of security may not guarantee absolute security when used in a real-world device or application. In parallel to spending research efforts to improve security of these APIs, we argue that it may be worthwhile to explore design criteria that would reduce the impact of an API exploit, assuming flaws cannot completely be removed from security APIs. We explore different solutions based on *salting* (i.e., using additional secrets with user PINs) and *localization* (i.e., service-point specific information) that may reduce the value of revealed PINs at intermediate switches in a banking network.

8.1 Introduction

Attacks on financial PIN processing APIs revealing customers’ PINs have been known to banks and security researchers for years, e.g., [54, 39, 41, 42, 40] (failure modes of ATM PIN encryption were first discussed in Anderson [9]). Apparently the most efficient of these *PIN cracking* attacks are due to Berkman and Ostrovsky [32].[1] However, proposals to counter such attacks are almost non-existent in the literature, other than a few suggestions; for example, maintaining the secrecy (and integrity) of some data elements related to PIN processing (that are considered security insensitive according

[1] We encourage readers unfamiliar with financial PIN processing APIs and PIN cracking attacks to consult Section 8.2 for background, and Appendix C for a summary of attacks by Berkman and Ostrovsky [32].

to current banking standards) such as the "decimalization table" and "PIN Verification Values (PVVs)/Offsets" has been emphasized [41, 32]. However, implementing these suggestions requires modifications to all involved parties' Hardware Security Modules (HSMs). Commercial solutions such as the `PrivateServer Switch-HSM` [8] rely mostly on *tightly* controlling the key uploading process to a switch and removing unnecessary APIs or weak PIN block formats. Even if the flawed APIs are fixed, or non-essential attack APIs are removed to prevent these attacks, it may be difficult in practice to ensure that all intermediate (third-party controlled) switches are updated accordingly. Thus banks rely mainly on protection mechanisms provided within banking standards, and *policy-based* solutions, e.g., mutual banking agreements to protect customer PINs.

MP-Auth (see Chapter 3) is apparently capable of preventing these attacks in addition to saving PINs from false ATM keypads and card reader attacks. However, MP-Auth relies on public key operations, and thus cannot be deployed without significant modifications to ATMs, switches and verification facilities. Another obvious solution (as suggested in [41, 32]) is to update the PIN processing APIs, which also requires modifications to all involved parties' Hardware Security Modules (HSMs).[2]

Designing solutions to mitigate PIN cracking attacks pose some interesting challenges. PIN transfers in banking networks rely on symmetric key cryptography where the third-party controlled intermediate switches also possess shared keys to decrypt encrypted PINs (but have no access to issuer/verification keys). Although decrypted PINs (and the decryption key itself) are not (ideally) accessible from outside of an HSM, API flaws allow attackers to realistically extract enough information from the HSM (through legitimate API calls) to enable PIN cracking attacks. Thus PIN cracking solutions must protect user PINs that travel through third-party switches which may be less security conscious or even actively malicious. Our solution attempts to address threats from such an adversary as well as hostile parties at a verification facility with limited access (e.g., one who can call API functions from an HSM, but cannot access verification keys). However, we do not consider ATM frauds that are not scalable such as false keypads and card reader attacks [71].

[2] For an overview of HSMs and related attacks, see Anderson et al. [10]

One primary reason that PIN cracking attacks are possible is that actual user PINs, although encrypted, travel from ATMs to a verification facility through several (untrustworthy) intermediate switches. If, for example, hashed PINs were sent in an encrypted form, attackers may not be able to reveal user PINs even in the presence of API flaws. However, as PINs are generally short (4 digits), an offline dictionary attack may still easily allow recovery of actual PINs. From reviewing the history of API attacks, we also note that even a complete overhauling of PIN processing APIs may be subject to presently-unknown API flaws that might be exploited to reveal user PINs. Therefore we seek a solution that precludes real user PINs being extracted at verification facilities, and especially at switches (which are beyond the control of issuing banks), even in the presence of API flaws. One possible solution in this direction is not to send the actual user PIN itself through untrusted intermediate nodes. Our proposal follows such a direction.

While PIN cracking attacks get more expensive as the PIN length increases, it is unrealistic to consider larger (e.g., 12-digit) user PINs, for usability reasons.[3] As part of our proposal, we assume that a unique random *salt* value of sufficient length (e.g., 128 bits) is stored on a user's bank card, and used along with the user's regular four-digit PIN (*Final PIN*) to generate[4] a larger (e.g., 12 digit) *Transport Final PIN* (TFP). This TFP is then encrypted and sent through the intermediate switches. Thus we essentially expand the 4-digit PIN to 12 digits. We build our *salted-PIN* solution on this simple idea. Our proposal requires updating bank cards (magnetic-stripe/chip card), service-points (e.g., ATMs), and issuer/verification HSMs. However, our design goal is to avoid changing any intermediate switches, or requiring intermediate switches be trusted or compliant to anything beyond existing banking standards.

Salted-PIN provides the following benefits.

1. It does not depend on policy-based assumptions, and limits existing PIN cracking attacks even where intermediate switches are malicious.

[3] A 12-digit PIN can be constructed by storing eight digits on the bank card while a user memorizes the other four digits as usual. However, as the real PIN is sent encrypted in this solution, attackers at a malicious switch can recover the PIN and create fake cards. (An anonymous FC 2008 referee pointed us to this idea and its relative advantages and disadvantages.)

[4] For example, through a pseudo-random function (PRF).

2. It significantly increases the cost of launching known PIN cracking attacks; for example, the setup cost for the translate-only attack (see Appendix C) for building a complete Encrypted PIN Block (EPB) table now requires more than a trillion API calls in contrast to 10,000 calls as in Berkman and Ostrovsky [32].
3. Incorporating service-point specific information such as "card acceptor identification code" and "card acceptor name/location" (as in ISO 8583) into variants of salted-PIN, we further restrict attacks to be limited to a particular location/ATM.

Organization. Background on financial PIN processing is provided in Section 8.2. We outline the proposed salted-PIN solution in Section 8.3. Known attacks that apply to the basic version of salted-PIN are discussed in Section 3.3.3. In Section 8.5 we introduce three variants of salted-PIN to counter these attacks. Implementation challenges are also briefly discussed in Section 8.5. Section 2.5 concludes. In Appendix C, we review several (representative) attacks from Berkman and Ostrovsky [32].

8.2 Background

In this section, we provide a basic overview of PIN processing and PIN block formats. More background on banking networks is discussed elsewhere (e.g., [54, 194]).

PIN processing architecture. When a user inputs her PIN at an ATM, the PIN is encrypted to form an Encrypted PIN Block (EPB) using a transport key shared between the ATM and the next switch connected to the ATM. A switch can be a stand-alone facility for PIN transportation (and other related bank network activities), or part of a bank's verification facility. PIN blocks are processed inside Hardware Security Modules (HSMs). Each switch shares a transport key with other switches that it is connected to. At a verification center, a switch may also have the issuer key (for PIN verification). A standardized set of PIN processing APIs is used for PIN creation, transportation, and verification. The intent is that this allows banks to protect user PINs from application programmers (or anyone having access to PIN processing APIs) at verification facilities as well as in switches.

There are several standardized PIN block formats (see below). An EPB may travel across several HSMs on its way to a verification site. When transmitted from one HSM to another, re-formatting (i.e., translating from one PIN block format to another) may be required. Thus all HSMs must implement translation APIs to allow reformatting of an EPB. A switch decrypts an EPB, checks the PIN block format (e.g., validity of PIN digits, PIN length), changes the format if required, and re-encrypts the PIN block with the destination switch's transport key. As all PIN operations are performed by HSMs, an application programmer (ideally) cannot learn anything about PINs transported as EPBs.

PIN Block formats. We outline four PIN block formats from ISO 9564-1 [127], three of which are approved by VISA for online transactions (e.g., through ATMs). Assume that a PIN is four decimal digits long. A PIN block is composed of 16 hex digits, i.e., 64-bits. Let 'P' be a PIN digit (0 to 9), PAN the least significant 12 digits of a customer's Primary Account Number (excluding the check digit), and let 'A' be a PAN digit (0 to 9). An ISO-0 PIN block is calculated as follows.

$$\begin{aligned}
\text{ISO-0 PIN Block} &= \text{Original PIN Block} \oplus \text{Formatted PAN} \\
\text{Here, Original PIN Block} &= \texttt{04 PPPP FFFF FFFF FF}, \\
&\quad \text{with 'F' denoting the hex digit F} \\
\text{Formatted PAN Block} &= \texttt{00 00AA AAAA AAAA AA}
\end{aligned}$$

The leftmost zero in the original PIN block stands for ISO-0, and the digit 4 is the PIN length (which could be as high as 12). An ISO-0 PIN block is the result of XORing an original PIN block with a formatted PAN. The ISO-1 PIN Block format is 14 PPPP RRRR RRRR RR, where 'R' is a random hex digit (0 to F). The ISO-2 PIN Block format is 24 PPPP FFFF FFFF FF, which is used only when creating a card. An ISO-3 PIN block is calculated as follows.

$$\begin{aligned}
\text{ISO-3 PIN Block} &= \text{Formatted PIN Block} \oplus \text{Formatted PAN} \\
\text{Here, Formatted PIN Block} &= \texttt{34 PPPP RRRR RRRR RR}, \\
&\quad \text{with 'R' a hex digit from A to F} \\
\text{Formatted PAN Block} &= \texttt{00 00AA AAAA AAAA AA}
\end{aligned}$$

In summary, ISO-2 is the weakest PIN format; it is not allowed for online processing, and it has not been used in the PIN cracking attacks. ISO-0 and ISO-3 PIN blocks depend on a user PIN and account number. ISO-1 format is not bound to a user's account number, and is recommended to be used in situations where the PAN is unavailable. Attacks exploiting translate-only APIs (see Section C.1) depend on the fact that any ISO-0 and ISO-3 PIN formats can be translated to the less secure ISO-1 format (as the ISO-1 format does not depend on the user PAN). Translation APIs are also generally implemented by all HSMs.

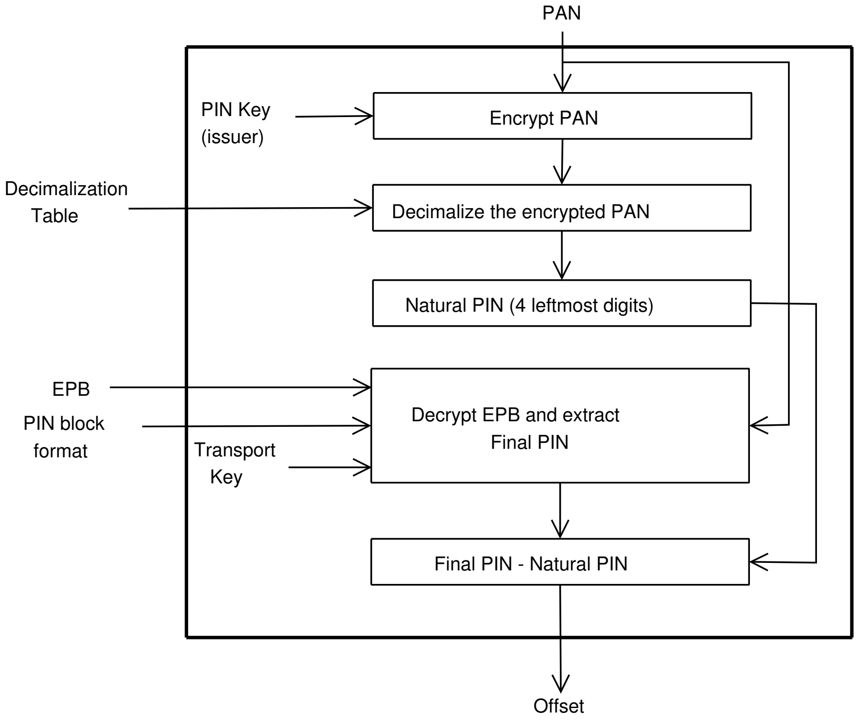


Figure 8.1: Offset calculation (adapted from [194])

IBM calculate-offset API. IBM's calculate-offset API outputs an offset using a PAN and EPB. If the calculated offset value corresponds to the stored value for that PAN, then the PIN inside the EPB is verified. Offset values are assumed by the banking standards to be security insensitive, and are generally stored in plaintext. Fig. 8.1 illustrates how an offset value is calculated for PIN verification. Here, a

Natural PIN is calculated from a customer's PAN, and the *Final PIN* is a customer-chosen PIN. Subtraction is digit by digit modulo 10. An issuer key (residing inside an HSM) is used to encrypt a user's PAN. The encrypted PAN may contain hex digits (A to F), and it is decimalized using a decimalization table (mapping hex digits to decimal digits). The four left-most digits of the decimalized encrypted PAN constitute the user's Natural PIN. The Final PIN is extracted from the user's PAN, EPB (containing the user's encrypted Final PIN), the PIN block format, and the transport key (residing inside the HSM). The offset is calculated by subtracting the Natural PIN from the Final PIN.

VISA PIN Verification Value (PVV). Fig. 8.2 depicts how a VISA PIN Verification Value (PVV) is calculated. PVVs are used in a similar fashion as IBM offset values, and also (generally) stored in a plaintext database. A customer's PVV may be written on her bank card as well (for offline PIN verification).

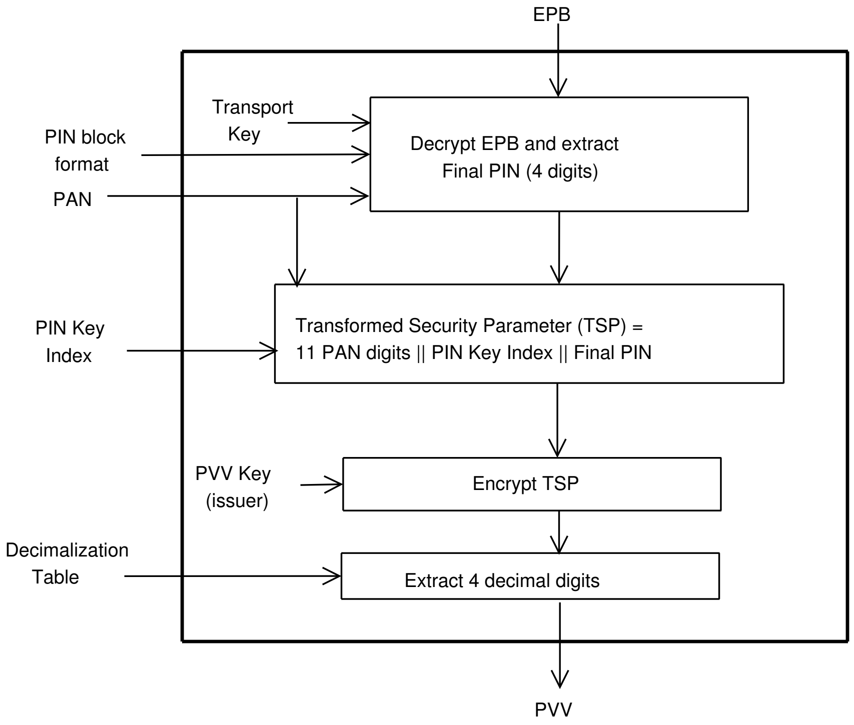

EPB
Transport Key
PIN block format
PAN
Decrypt EPB and extract Final PIN (4 digits)
PIN Key Index
Transformed Security Parameter (TSP) = 11 PAN digits || PIN Key Index || Final PIN
PVV Key (issuer)
Encrypt TSP
Decimalization Table
Extract 4 decimal digits
PVV

Figure 8.2: PVV calculation (adapted from [194], || denotes concatenation)

8.3 Salted-PIN

Here we present the salted-PIN proposal in its simplest form.

Threat model and notation. Our threat model assumes attackers have access to PIN processing APIs and transaction data (e.g., Encrypted PIN Blocks, account number) at switches or verification centers, but do not have direct access to keys inside an HSM, or modify HSMs in any way. Attackers can also create fake cards from information extracted at switches or verification centers and use those cards (perhaps through outsider accomplices). We primarily consider large scale attacks such as those that can extract millions of PINs in an hour [32]. We do not address attacks that are not scalable, such as *card skimming*, attacks on EMV[5] PIN entry devices [71], or cases where an accomplice steals a card and calls an insider at a switch or verification center for an appropriate PIN. PIN cracking attacks that we consider are successful only when online PIN verification is applied (i.e., encrypted PINs are sent to a verification center for approval). In addition to magnetic-stripe cards, these attacks are also valid for chip/EMV cards except when offline/on-chip PIN verification is used (assuming card issuers allow EMV cards to fallback to magstripe processing for backward compatibility or chip failure). We use notation from Table 8.1.

PAN	User's Primary Account Number (generally 14 or 16-digit).
PIN	User's Final PIN (e.g., 4-digit, issued by the bank or chosen by the user).
PIN_t	User's Transport Final PIN (TFP).
$Salt$	Long-term secret value shared between the user card and issuing bank.
$f_K(\cdot)$	A cryptographically secure Pseudo-Random Function (PRF).[6] Here K is the PRF key.

Table 8.1: Notation used in Salted-PIN

Note that, if and when chip cards are deployed worldwide, and offline PIN verification is the de facto mode of operation, most current PIN cracking attacks involving

[5] EMV is a growing standard for chip-based bank cards, initially developed by Europay, MasterCard, and VISA; see `http://www.emvco.com`.

[6] For example, as used in `PwdHash` [214].

intermediate switches become invalid, consequently eliminating the need for new solutions. Although such cards are being gradually adopted, apparently magstripe cards (and the magstripe mode of operation of chip cards) along with existing flawed APIs will remain in operation for a long time to come. For example, we have seen that the transition away from DES and triple-DES to more modern cryptographic algorithms has taken far longer than many might have originally predicted.

Generating salted-PINs. A randomly generated salt value of adequate length (e.g., 128 bits, to make dictionary attacks infeasible) is selected by a bank for each customer. The salt is stored on a bank card (chip-card or magstripe) in plaintext, and in an encrypted form at a verification facility under a bank-chosen *salt key*. API programmers (i.e., those who use HSM APIs) have access to this encrypted salt (but do not know the salt key or plaintext salt values). Encrypted salt values also cannot be overwritten by API programmers. A user inputs her PIN at an ATM, and the ATM reads the plaintext salt value from the user's bank card and generates a Transport Final PIN (TFP) as follows.

$$PIN_t = f_{Salt}(PAN, PIN) \tag{8.1}$$

The PRF output is interpreted as a number and divided by 10^{12}; the 12-digit remainder (i.e., PRF output modulo 10^{12}) is chosen as PIN_t and treated as the Final PIN from the user. Note that the maximum allowed PIN length by ISO standards is 12. The ATM encrypts PIN_t with the transport key shared with the adjacent switch, and forms an Encrypted PIN Block (EPB). An intermediate switch decrypts an EPB, (optionally) reformats the PIN block, and re-encrypts using the next switch's transport key. Additional functionalities are not required from these switches.

To set the initial offset or PIN verification value (PVV), an issuer generates a random PIN (e.g., 4 digits long) and salt for a user, and then uses equation (8.1) to generate PIN_t. The transport key of the verification HSM is used to encrypt PIN_t and form an EPB. This EPB is used to call a calculate offset/PVV function with the user's PAN and encrypted salt to generate the initial offset/PVV (note that each of these values is now 12 digits long).

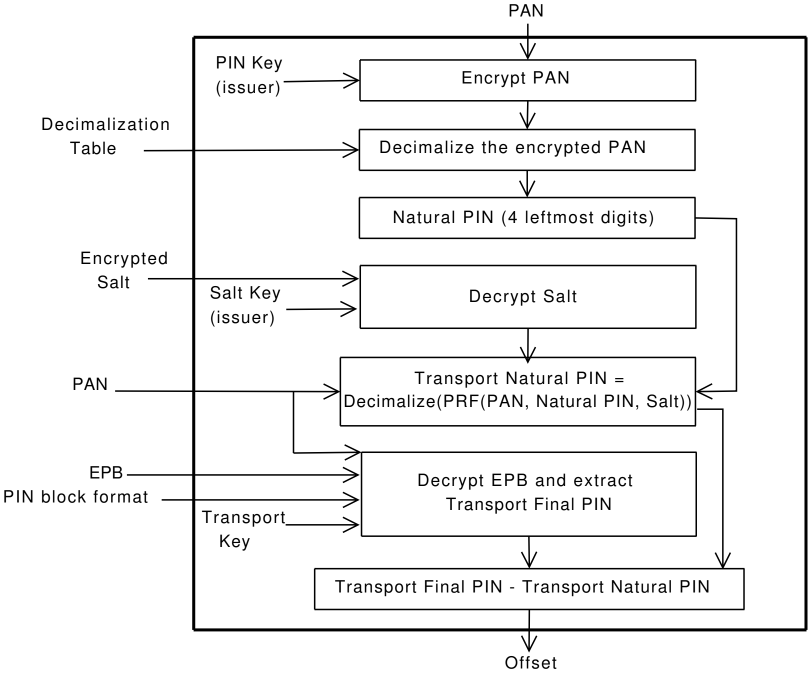

PAN
PIN Key (issuer)
Encrypt PAN
Decimalization Table
Decimalize the encrypted PAN
Natural PIN (4 leftmost digits)
Encrypted Salt
Salt Key (issuer)
Decrypt Salt
PAN
Transport Natural PIN = Decimalize(PRF(PAN, Natural PIN, Salt))
EPB
PIN block format
Transport Key
Decrypt EPB and extract Transport Final PIN
Transport Final PIN - Transport Natural PIN
Offset

Figure 8.3: Salted-PIN verification for the IBM offset method

Offset/PVV verification with salted-PIN. The salted-PIN verification for the IBM offset method (recall Section 8.2) is shown in Fig. 8.3. The Natural PIN is calculated from a PAN using an issuer's PIN key. The encrypted salt value corresponding to the PAN is decrypted using a salt key (like the PIN key and transport key, the salt key also resides inside an HSM). The Transport Natural PIN is generated from the Natural PIN using equation (8.1). The Transport Final PIN is extracted from an EPB, and the Transport Natural PIN is subtracted from it (digit by digit modulo 10 subtraction) to get the offset. This calculated offset value is compared with the corresponding PAN's stored (e.g., in a database) offset value. The salted-PIN verification for VISA PVV is shown in Fig. 8.4. The salt value is appended at the end of the Transformed Security Parameter (TSP), which is encrypted and decimalized to calculate the PVV. Note that we design the offset/PVV verification functions to keep them similar to the existing functions although these can be further simplified; for example, instead of storing offset/PVV values, EPBs directly may be stored and compared with incoming EPBs.

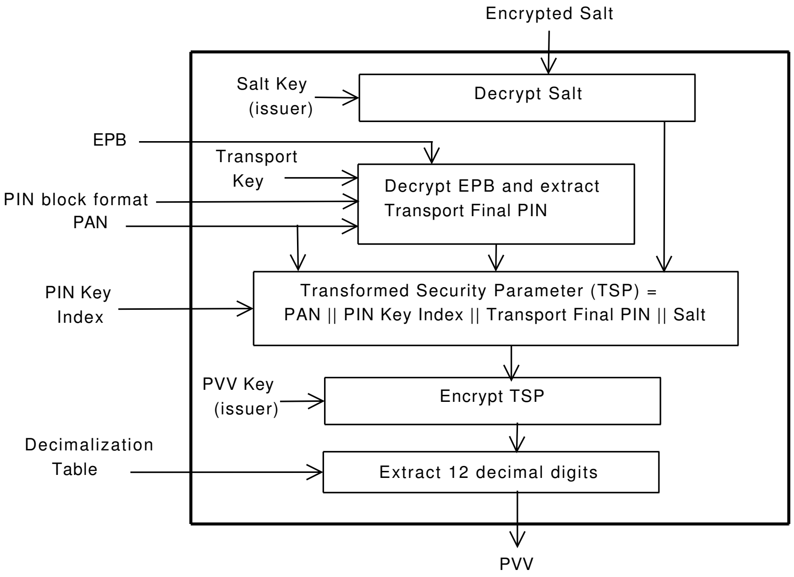

Encrypted Salt
Salt Key (issuer)
Decrypt Salt
EPB
Transport Key
PIN block format
PAN
Decrypt EPB and extract Transport Final PIN
PIN Key Index
Transformed Security Parameter (TSP) = PAN || PIN Key Index || Transport Final PIN || Salt
PVV Key (issuer)
Encrypt TSP
Decimalization Table
Extract 12 decimal digits
PVV

Figure 8.4: Salted-PIN verification for the VISA PVV method

Salted-PIN protection against PIN cracking attacks. We discuss attacks (e.g., translate-only [32]) that reveal a user's TFP in Section 8.4. An attacker with write-access to the PVV database at a verification facility can choose any PIN for a specific account (see Section C.3). With the salted-PIN solution, an attacker can still choose any PIN to pack in an EPB and write the resulting PVV to a database. However, without knowing the salt value, overwriting a user's PVV does not help in an attack for the following reason. The salted-PIN verification function for PVV (Fig. 8.4) ensures use of the encrypted salt value as indexed by a user's PAN; thus for a successful PVV verification, a user's salt must be known or the encrypted salt value must be replaced.

8.4 Attacks on Salted-PIN

We now discuss attacks against the basic version of salted-PIN.

8.4.1 Enumerating EPBs through Translate-only Attacks

Here the goal of an attacker is to create a table of EPBs, and then crack all or a subset of user accounts. The following attacks in part follows an efficient variant of the translate attack as outlined by Berkman and Ostrovsky [32]. For these attacks, we assume an attacker M_i is an insider (e.g., application programmer) at a switch or verification center, and an outsider accomplice M_a who helps M_i in carrying out user input at an ATM. These attacks are possible for the following reason. Although a TFP is calculated from a long (e.g., 128 bits, sufficient to deter dictionary attacks) salt value, only 12 digits of the PRF output are used. Thus an attacker only requires *any* pair of salt and PIN combination that can generate a targeted account's TFP instead of finding the actual salt/PIN values.

Targeting all accounts. Assume that M_i extracts the salt value ($Salt_a$) and PAN from a card he possesses, and uses equation (8.1) to generate the 12-digit TFP PIN_{at} (through software or a hardware device, using any PIN PIN_a). Let PIN_{at} consist of $p_1p_2p_3 \ldots p_{12}$ where each p_i ($i = 1$ to 12) is a valid PIN digit. Then M_a inserts this card to an ATM, and enters PIN_a. Assume that the generated PIN_{at} is encrypted by the ATM to form an EPB, E_1. M_i captures E_1 at a switch. If E_1 is not in the ISO-1 format, M_i translates it into ISO-1 (to disconnect E_1 from the associated PAN). Let the translated (if needed) E_1 in the ISO-1 format be E_1'. E_1' is then translated from ISO-1 to ISO-0 using $p_3p_4 \ldots p_{12}00$ as the input PAN. This special PAN is chosen so that the XOR of PIN positions 3 to 12 with PAN positions 1 to 10 removes $p_3 \ldots p_{12}$ when the translation API is called; i.e.,

PIN block inside E_1' =	0	C	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	F	F
Input PAN =	0	0	0	0	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}	0	0
Resulting ISO-0 PIN block =	0	C	p_1	p_2	0	0	0	0	0	0	0	0	0	0	F	F

Assume the resulting EPB is $E_{p_1p_2}$ which is the same as the one containing a TFP $p_1p_2$0000000000 with PAN 0. Now we can create all EPBs containing every 12 digit TFPs starting with p_1p_2 from $E_{p_1p_2}$. For example, an EPB with $p_1p_2q_3q_4 \ldots q_{12}$ as the TFP can be generated through transforming $E_{p_1p_2}$ using PAN $q_3q_4 \ldots q_{12}00$ (in ISO-0). Thus we can create all 10^{10} EPBs with TFPs from $p_1p_20 \ldots 0$ to $p_1p_29 \ldots 9$.

Starting from a different p_1p_2, all 10^{12} EPBs containing every 12 digit TFP can be generated as follows. M_a uses the previous bank card (i.e., the same salt and PAN) with different PINs (obviously, including wrong PINs) to calculate TFPs using software or a special device. When a TFP is found with the first two digits different than p_1p_2, the corresponding PIN is entered at an ATM. The attacker M_i at the switch then generates another set of 10^{10} EPBs containing TFPs starting with this different p_1p_2. The attack continues with different PINs until all 100 possible values of the initial two TFP digits are covered. Thus using these 100 EPBs containing TFPs starting with the different first two digits (i.e., from 00 to 99), M_i can create a table of EPBs for all possible TFPs (with corresponding PINs). The cost of building this table is slightly over 10^{12} API calls (for each 100 $E_{p_1p_2}$, at most two API calls are required). The cost of selecting the initial EPBs (i.e., that contain TFPs with two different starting digits) is insignificant as M_a can calculate TFPs offline, i.e., without involving any API calls to HSMs.

To launch an attack, a valid EPB of a target customer is collected. The EPB is translated to ISO-1 (to decouple it from the target account, if not already in ISO-1), then to ISO-0 with PAN 0. The resulting EPB is then located on the EPB table (as created in the setup phase). The corresponding PIN from the table can now be used to exploit a card generated with the target's PAN, and the attacker's salt value (i.e., $Salt_a$). The cost of this attack is at most two API calls and a search of $O(10^{12})$, i.e., $O(2^{40})$.

In summary, the setup cost of this attack is about 10^{12} API calls with a per account cost of two API calls plus a search of $O(10^{12})$. The same translate-only attack by Berkman and Ostrovsky [32] on the current implementation of PIN processing requires only about 10,000 API calls as setup cost, and a per account cost of two API calls plus a search of $O(10^3)$.

Trade-off between table size and per EPB attack cost. The per account cost of the above attack is not high enough to deter an attack. However, the setup cost of building the table with all one trillion EPBs is apparently significant (although this is a one-time cost). By reducing the table size, the attack can be launched with fewer API calls although the per EPB attack cost increases accordingly.

Assume that the attacker builds a table of 10^6 EPBs (i.e., one half of the original table size) containing TFPs ending with six zeros (000000), i.e., storing only the first six digits of a TFP. With this table, an attacker can calculate TFP of any target EPB E_c in 10^6 steps (assuming the EPB arrives in ISO-1 format, or the attacker translates it into ISO-1); each step then requires one API call. The attack is described in Algorithm 1.

Algorithm 1 Steps in the partial table attack

```
for $i = 0$ to $10^6 - 1$ do
    $E_{c0} = \mathsf{Translate}_{\mathsf{ISO-0}}(E_c, i \times 100)$
    if $E_{c0}$ is in the table then
        TFP in $E_c = 10^6 \times$ (six digit TFP from the table) $+ i$
        Salt and PIN values corresponding to $E_c$ is used to generate a fake card
        exit
    end if
end for
```

Now the cost of attacking N accounts is $10^6 + N \times 10^6$ API calls. The attacker can also vary the table size and the number of steps for each target account. For any table size 10^n for $n \in \{2, 3, \ldots, 12\}$, the required number of per account translate steps is 10^{12-n}. Thus in general the cost of attacking N account is $10^n + N \times 10^{12-n}$.

8.4.2 Replay Attack

In this attack, an adversary M_i at a switch or verification center collects a valid EPB E_c for a target PAN A_c, and then creates a fake card with the account number A_c (and any salt value). Note that M_i here does not know the actual salt value or PIN for the target account. An accomplice M_a uses the fake card with any PIN at an ATM, and the ATM generates a false EPB E_a. At the switch/verification center M_i locates E_a in transfer, and replaces E_a with the previously collected correct EPB E_c. Thus the fake card will be verified by the target bank, and M_a can access the victim's account.

Note that this attack works against the basic variant of salted-PIN as well as current PIN implementations without requiring any API calls. Although quite intuitive, this attack has not been discussed elsewhere to our knowledge.

8.5 Variants, Implementation Challenges and Lessons Learned

As we discussed in Section 8.4, the basic version of salted-PIN is vulnerable to several attacks. Other than the replay attack, the setup cost of launching these attacks is not trivial as previous PIN cracking attacks (cf. [32]) although the per account attack cost is apparently manageable. In this section, we outline three variants of salted-PIN to practically restrict these attacks by increasing the per account attack cost.

Variant 1: Localized / service-point specific salted-PIN. If a fake bank card is created for a target account (e.g., through the attacks in Section 8.4), the card can be used from anywhere as long as it remains valid (i.e., the issuing bank does not cancel it). To restrict such attacks, we modify equation (8.1) as follows.

$$PIN_t = f_{Salt}(PAN, PIN, spsi) \tag{8.2}$$

Here *spsi* stands for *service-point specific information* such as a "card acceptor identification code" and "card acceptor name/location" as in ISO 8583 (Data Elements fields). The verification center must receive *spsi* as used in equation (8.2). Although any PIN cracking attack (Section 8.4.1) can be used to learn a TFP or build a full/partial EPB table, the table is valid only for the particular values of *spsi*. Also, the replay attack (Section 8.4.2) may succeed only when the accomplice exploits a compromised card from a particular ATM. Thus this construct generates a localized TFP for each PIN verification, and thereby restricts the fake card to be used only from a particular location/ATM. Note that for this variant, the verification facility cannot use PVV or Offset values, because they would be different for each ATM. Another verification value would need to be designed.

Variant 2: Salted-PIN with double EPBs. ISO PIN block formats restrict PIN length to 12 digits in an EPB. This length limit enables a search of $O(2^{40})$ in a prebuilt table (see in Section 8.4.1). As a variant, instead of choosing 12 digits from the result of equation (8.1), we can take 24 digits (i.e., PRF output modulo 10^{24}) and create two PIN_t blocks, each 12 digits long. As a result, two EPBs must be sent from an ATM, and a verification facility needs both EPBs to verify a user's PIN. However, intermediate switches may not need to be aware of this. An attack similar to Section 8.4.1 can be launched on each EPB separately, and two tables can be built for both parts of a 24-digit TFP; the cost of building the table simply doubles (two TFP tables, each has 10^{12} entries). Using the tables, a 24-digit TFP can be extracted from the two EPBs of any target account. However, determining a valid pair of salt value and PIN is not straightforward as the attack in Section 8.4.1. To generate a fake card (i.e., to find an appropriate salt value and PIN for the intended TFP) for this variant of salted-PIN, attackers must apparently carry out a computation of 10^{24} (i.e., $O(2^{80})$) steps. However, this variant is vulnerable to the replay attack (Section 8.4.2) when equation (8.1) is used. Again, service-point specific information as used in equation (8.2) for generating TFP can practically limit such attacks.

Variant 3: End-to-end PIN encryption/MAC. Using the stored salt as an encryption key, end-to-end PIN encryption can be achieved between an ATM and verification center. The salt value can also be used for calculating a message authentication code (MAC) for a user's Final PIN. This variant can secure PIN transportation to the extent of the algorithm used for encryption or MAC. Thus it can effectively eliminate PIN enumeration by an attacker at a switch or verification center. However, to restrict the replay attack (Section 8.4.2), one or more service-point specific items must be used with a PIN for encryption or MAC. Also, this variant will require updating intermediate switches.

Implementation challenges. One implementation challenge for salted-PIN could be the storage requirement for the salt (39 decimal digits or 128 bits) that must be stored on a bank card. There are four possible scenarios: (1) magnetic-stripe (magstripe) cards; (2) chip-card with a magnetic stripe at a magstripe reader terminal; (3) chip-card with online PIN verification; and (4) chip-card with offline PIN verification. For the last case, as a PIN does not leave the card, PIN cracking attacks are immaterial. For the first two cases, the amount of data that can be stored on a magnetic stripe is limited by ISO standards; for example, according to ISO-7811, track one in a magstripe bank card holds 79 six-bit characters (plus a parity check), and track two holds 40 four-bit (plus a parity) characters. These two tracks are generally present in most magstripe bank cards (there is also a third track on some cards). A salt may be stored on a magstripe card by overloading non-essential data fields in track one (e.g., discretionary data, name, expiration date), and redundant fields in track two (e.g., PAN). Chip-cards offer significantly more storage capability, and thus for the third case, accommodating the salt may not be an issue.

Salted-PIN requires that service points (e.g., ATMs, point-of-sale terminals) are capable of computing PRF as in equation (8.1). Thus another implementation challenge is posed by the limited computing ability of old magstripe reader terminals with limited CPU capabilities and cryptographic support of only a DES chip; recent terminals (e.g., Motorola's PD4750) generally operate on a 32-bit processor, and computing a PRF is not a computational issue.

Lessons learned and discussion. Now we briefly discuss the lessons learned from designing different variants of salted-PIN. These lessons, we believe, may help others in building robust security protocols.

1. **Minimizing disclosure of reusable information.** In the banking network, encrypted user PINs are sent through multiple switches to the verification center for user authentication. Such a scheme always bears the risk of exposing the long-term, reusable secret PIN. We argue that if long-term secrets are converted to one-time use passcodes, then the discovery of a flaw may not necessarily lead to the compromise of a reusable secret. Some techniques such as Lamport's hash chain [147] have been publicly known for decades. Unfortunately, applications of these schemes appear to be low.

2. **Reducing the value of disclosed information.** In general, currently attackers enjoy the benefit of compromising sensitive secrets once, and then reusing those multiple times. Localization of secrets or sensitive information as applied in the service-point specific salted-PIN variant, may also be useful in other settings, e.g., restricting identity fraud as a result of data breaches (see Chapter 7). Making attacks *unattractive* (i.e., the reward is less than the required efforts) is an easier goal than making attacks *impossible*, and is often effective and sufficient. As defenders (such as API designers) and attackers are both humans, it makes little sense, at least on a philosophical ground, to believe that defenders can design protocols or techniques that cannot be defeated one way or another. However, we can *design for damage control*, i.e., design protocols in such a way that when they break, they still do not expose long-term user secrets. Incorporating such damage control techniques into the design itself may make our protocols more resilient to attacks.

8.6 Concluding Remarks

In the 30-year history of financial PIN processing APIs, several flaws have been uncovered. Here we summarize some API attacks from Berkman and Ostrovsky [32] for context, and introduce a *salted-PIN* proposal and three of its variants to counter these attacks. Our preliminary analysis indicates that salted-PIN can provide a higher barrier to these attacks in practice by making them considerably more expensive (computationally). We have discussed some deployment issues, but acknowledge that this discussion is not exhaustive; deployment barriers may arise from unseen aspects. Salted-PIN is motivated primarily by the realistic scenario in which an adversary may control switches, and use any standard API functions to reveal a user's PIN; i.e., an attacker has the ability to perform malicious API calls to HSMs, but cannot otherwise modify an HSM.

Our proposal of salted-PIN is intended to stimulate further research and solicit feedback from the banking community regarding: (1) whether salted-PIN may improve PIN security in real terms; (2) practical barriers of deploying salted-PIN; and (3) any significant weaknesses of salted-PIN. We focus on providing a technical solution to update PIN processing APIs, some of which are well-known to be flawed. Instead of relying, perhaps unrealistically, on honest intermediate parties (who diligently comply with mutual banking agreements), we strongly encourage the banking community to invest effort in designing protocols that do not rely on such assumptions which end-users (among others) have no way of verifying. It has been speculated [32] that PIN cracking attacks may explain numerous unexplained *phantom* withdrawals [38] as reported by many ATM fraud victims. As reported [281] recently (June 20, 2008), the compromise of a third-party PIN processor may have been the reason for a large number of Citibank card fraud.

Chapter 9

Comparative Summary and Concluding Remarks

In this chapter, we provide a comparative summary of our proposals, discuss the relationship among our different threat models, and revisit the thesis objectives. We also list a number of lessons learned in the course of this work which may help others to design better tools for addressing real-world security issues. Several open problems related to this thesis are also discussed.

9.1 Threat Models and their Justification

Our proposed techniques for addressing several different real-world problems have different but related threat models. Below we provide an overview of these models.

The threat model for MP-Auth includes compromised user PCs and attacks on users' mental models of the web. However, we assume the user device (cellphone or a stand-alone mobile/personal device) as used in MP-Auth is malware-free. If commonly used cellphones cannot be made secure (in the sense of free of malware), we argue that a separate device with minimum configuration must be designed; this device may be integrated with other devices that users generally carry. Recently, others also have envisioned such a device; see e.g., [148]. We also assume that users will not share their passwords for MP-Auth-enabled services with any other applications or sites, and input these passwords only through a device. Additionally, we expect users to be vigilant while confirming a transaction through their device. MP-Auth does not address the privacy threat of malware on the PC having read-only availability of user transactions. The loss of transaction privacy is no less important, but we believe that protecting users' long-term credentials is more critical, at least in the current environment.

It is quite plausible that users' MP-Auth passwords may be compromised, or users may be duped to confirm unwanted transactions. Our proposals for integrity verification (Section 4.2) address such threats. Also, to help users choose *strong* passwords, we introduce the Object-based Password (ObPwd) scheme which provide presumably much better entropy than most user-chosen passwords. We believe that these passwords can significantly reduce threats from online dictionary attacks on MP-Auth. ObPwd addresses the fact that at least some users will choose *weak* passwords, and that imposing password rules has been a failure in changing user choices so far. However, ObPwd assumes that the generated passwords will be used in a malware-free environment (i.e., no *rootkit* or equivalent), and that users will not choose obvious online objects as their password objects. Our hope is that as users own or have access to more and more (often portable) digital content, they have better chance of choosing a better password with ObPwd than current practices. Ideally, we also expect these passwords to be used from a trustworthy device with an MP-Auth-enabled service.

In IMPECS, we assume the threat to users' personal web content arises from the easy and universal availability of such content. Compromised hosts also pose a large threat to privacy, but we focus on current sharing mechanisms which are the primary cause of privacy breaches now. Also in a variant of IMPECS, we address the compromise of web servers hosting personal content. Our implementation of IMPECS relies on the security of underlying IM services or any other mechanisms used for distributing authentication *tokens*.

The threat model for ID number localization variants primarily focus on breached databases of sensitive identity numbers. Once compromised, these numbers can be exploited many times, in many different places. Instead of compromised user PCs, our focus here is compromised or lost user databases from relying parties (i.e., those who make use of ID numbers). We assume ID issuers can secure their data, or at least the probability of these parties being compromised is much less than relying parties mainly because the number of issuers is significantly less than that of relying parties. For PIN localization, the focus is to address threats from potentially malicious intermediate parties while we assume end-hosts such as ATM machines and verification sites are largely trustworthy. Unlike the SSL-protected Internet communication, banking networks allow intermediate nodes to decrypt and re-encrypt PIN blocks (due to the reliance on symmetric-key cryptography). We ignore threats from lost or stolen bank cards as attacks exploiting such cards are not scalable.

Summary and comparison. In summary, our threat models include compromised user PCs to content hosting servers to corporate and government databases. This wide range of attack targets is an indication of the variety of attacks faced by different real-world applications. In order for security techniques to have any real impact, we believe that more effort must be spent on identifying the varying landscapes of threats, and that these must be addressed in a pragmatic fashion.

We provide a comparative overview of our proposals in Table 9.1. On the left-most column, we list our proposals and on the horizontal heading, we list different protections and security features. Here we briefly describe the items under the top row. "Compromised host" covers user machines infected with any sort of malware – permanently installed (e.g., rootkit), or ephemeral (e.g., JavaScript keyloggers). Semantic attacks on regular users are considered under "Phishing." "Data breaches (misuse)" accounts for the exploitation of leaked personal credentials. "Commit unauth. trans." stands for the threat of committing unauthorized transactions and "Detect unauth. trans." is for detecting unauthorized transactions. The column labelled "Privacy" indicates whether privacy of user actions (performed with long-term credentials) is protected under a certain proposal. "Long-term credential protected" covers whether a user's long-term (reusable) account credentials are protected by a specific technique.

A (✓) means a security protection/feature is provided, and an (✗) means the lack of that protection/feature. NA denotes non-applicability. (All ✓and no ✗ would be optimal.) For example, MP-Auth provides protection against compromised hosts, phishing and committing unauthorized transactions, and safeguards long-term credentials; but MP-Auth cannot detect unauthorized transactions committed beyond an MP-Auth-protected session, restrict misuse of breached credentials, or provide transaction privacy. We acknowledge that although this table may provide useful high-level overview, this is not comprehensive (e.g., not all protections/features are included in the table). Details of these protections and security features are provided in relevant chapters. Also note that, even though we listed IMPECS and localized

PIN schemes as not providing protection against compromised hosts, they can easily be incorporated with an MP-Auth-like mechanism for credential input, and thus largely be made immune to threats from compromised hosts.

	Threats addressed				Features enabled		
	Compromised host	Phishing	Data breaches (misuse)	Commit unauth. trans.	Detect unauth. trans.	Privacy	Long-term credential protected
MP-Auth	✓	✓	✗	✓	✗	✗	✓
Integrity	✓	✓	✓	✗	✓	✗	✗
IMPECS	✗	✗	✓	NA	NA	✓	NA
Localized ID	✓	✗	✓	✓	✓	✓	✓
Localized PIN	✗	✗	✓	✓	✗	NA	✓

Table 9.1: Summary of our proposals with respect to threats addressed. These proposals have been discussed in previous chapters/sections: MP-Auth in Chapter 3, integrity verification techniques in Section 4.2, IMPECS in Chapter 5, localized ID in Chapter 7, and localized PIN in Chapter 8.

9.2 Recapping Thesis Objectives

Problem areas that we address include: (i) compromised hosts; (ii) semantic attacks; and (iii) repeated misuse of breached credentials. That each by itself is a large problem area is one reason why real-world security is so difficult. However, our focus is to explore example instances of real-world security issues, and show that protocols and techniques can be designed with more realistic assumptions than most academic proposals. We do not claim to have completely *solved* security and privacy problems in our proposals. Ideally, we would like to use these proposals to help reduce the disconnect between academic proposals and threats faced by millions of everyday computer users.

Even though the current untrusted environment is a challenge to work with, but we argue that our example proposals indicate an affirmative answer to *Question 1* in this thesis ("Can we design instances of technologies that can improve security and privacy in real-world applications, given the current state of compromised computing environment?"). We do not provide any *proof* for such a claim, but rely on the specific solutions that we consider. For example, MP-Auth can enable non-expert users to *safely* perform sensitive online tasks such as banking, considering several wide-spread threats including keylogging and phishing. IMPECS improves privacy of personal content on the web. Localized ID proposals address large-scale exploitation of compromised credentials. See Section 9.1 for more related discussion.

As for *Question 2* ("What would be the design criteria of such solutions? Can we suggest any general guidelines?"), even our limited number of proposals suggest that generalizing design principles is not straightforward. Solutions to some threats may be generalized, e.g., threats from keyloggers may be reduced if user input is provided through a separate device. However, compiling a comprehensive a list of common criteria for better protocol design seems elusive. We provide our design suggestions in Section 9.3.

We also set usability as one of our foci in this thesis. We begin with a questionnaire-based survey of online banking users. There have been numerous user studies on different security and privacy tools (e.g., PGP [278]), and specific online tasks (e.g., money transfer [121]). However, instead of considering any particular security tool or task, we focus on overall system security as deemed required by current (banking) practices for doing sensitive online tasks. Our user study apparently provides a better understanding of current requirements as expected from regular non-expert users. Among our proposals, ObPwd is informally tested by real users, and usability seems acceptable as reported in various online forums and the feedback received from users (many of whom may be technically-inclined). However, for other proposals we did not conduct any formal user study within the scope of this thesis. Such studies may be carried out in the future, and they may uncover usability drawbacks as currently unknown to us. To support usability, we rely on the *simplicity* [139] of our mechanisms, *familiarity* of additional steps or devices used, and by keeping the *number of steps* as low as possible. Note that some variants of our ID number localization and salting/localization of banking PINs proposals are transparent to end-users, and thus mandate no user testing.

As for restricting threats from the repeated use of compromised credentials, we explore two widely exploited and practical problems: breached personal records from service providers' databases and banking PINs from intermediate switches. While we cannot claim that our proposals are optimal, they clearly show that indeed using existing crypto tools we can design solutions to reduce the burden of defenders of such information.

9.3 Lessons Learned

Below we summarize important lessons from this thesis which we believe can help protocol designers achieve more effective and realistic security and privacy goals in the current untrusted environment.

Minimize requirements for additional security software. Our online banking user study shows that simply imposing (software) security requirements on non-expert users and expecting them to follow such *best practices* has no hope of success in any practical sense, other than perhaps some (unknown) advantage in shifting liability. Some of these requirements/steps (e.g., up-to-date patching) may be automated to a certain extent in a corporate/government environment which is often managed by experienced professionals. For home users, who make up the largest portion of online users, it is better not to make their security dependent on running any additional pieces of software programs; i.e., users should not be expected to install and maintain anything that does not accomplish any direct user tasks. Software, especially security software, cannot be easily maintained which is apparent from the fact that most corporate/government/educational networks are maintained by expert administrators, not by users themselves. As an example incident [258], many users have been deceived to install malware when the malicious program claimed to be *free* anti-virus software. When so many users do not even understand the difference between malware and anti-virus, it is a fallacy to require them to maintain/run security software.

Minimize the number of security-sensitive cognitive tasks. Users' mental models will remain vulnerable to innovative attacks, i.e., phishing is not going to go away anytime soon. Even if people become more familiar with technologies and deception techniques used by cybercriminals, we expect that threats from such attacks will always be significant. One wide-spread belief among security designers is that users can be *educated* over time. In the HCI community, which has a longer period of experience to draw on, it is now commonly accepted that user education is over-stated and often misused as the solution to many problems. Users may become more aware with time, but attackers are also evolving their techniques to take advantage of the new environment. Eliminating current security problems through unrealistic reliance on user education appears to be chasing a mirage. We believe that our best bets are: (i) to reduce the number of security related tasks as much as possible, so that the number of mistakes is reduced; and (ii) to deploy different layers of security, so that users can recover from their mistakes.

Use existing crypto tools *appropriately.* Crypto techniques have advanced significantly in the past thirty years. However, similar progress is less apparent in security tools. Some may argue that security is *difficult* to achieve as there always remains the possibility of implementation bugs or (unknown) inherent design flaws. We believe that available crypto tools are generally under-utilized to advance real-world security. For example, despite their mathematical elegance and seemingly obvious benefits, techniques based on public key cryptography have rarely been utilized to deliver meaningful security for common applications. However, existing crypto protocols and techniques are inadequate for achieving practical security as many threats faced by real-world applications are considered out-of-scope in many traditional crypto threat models. Our proposed protocols heavily rely on existing crypto techniques, but we tweaked those techniques for the current untrusted environment. These protocols are designed with *weaker* assumptions which take practical threats into consideration. We argue that crypto tools remain valuable for improving real-world security problems, but must be used differently than traditionally used by crypto experts.

Use non-crypto tools and techniques. Security researchers can evolve their techniques according to widely available tools and infrastructure. In addition to traditional crypto tools such as public key encryption, hashing, and message authentication code, we also involve personal devices and digital media files, cross-checking, IM networks, and existing mag-stripe/chip cards in our proposed protocols. This makes our designs somewhat *hybrid*, i.e., a mix of applied crypto and existing mechanisms. We believe that in the future, e.g., ten years from now if we re-design our proposals, we should make use of whatever technologies/infrastructure are available at that time. The lesson here is that security techniques must evolve and take advantage of other tools of the time.

Design for damage control. We believe that security designers must realize and accept that attackers are also smart and intelligent human beings, with a wide variation of resources. We argue that technologies should be designed with a *gracious* failure mode, i.e., one that limits harms done to users even when attackers win in the arms-race. It is interesting to note that making the defenders' failure less catastrophic may adversely affect the attackers. Bullet-proof or fool-proof technologies are difficult to achieve, and even more difficult to maintain for a long time. Technologies that restrict attackers and limit (rather than eliminate) damages are easier to design and far more cost-effective and realistic to achieve. As humans, it hurts our egos to accept that our mathematically-proven design will fail at some point in time. We argue that accepting such failures and designing our systems accordingly to reduce the damage will help improve overall real-world security.

Simplicity for easy deployment. In applications involving software, especially software that evolves and is re-implemented by many parties, simple techniques have much more chance in getting acceptance in practice than complex tools, even if the later ones provide better security guarantees. Most academic proposals aim for *ideal* security, resulting in designs too complex to be deployed in reality. In contrast, we focus on simple techniques based on existing tools and infrastructure, which apparently improve real-world security, and thus breaking the current status quo. Achieving incremental practical security should be prioritized over theoretical finesse, if the goal is to impact security in the real world.

9.4 Open Problems

In this thesis, we consider several real-world threats and provide solutions to specific problem instances. Below we briefly discuss certain open problems related to our proposals.

We design MP-Auth for addressing threats from malware in user PC and phishing. MP-Auth protects a user's sensitive login credentials being captured by malware, and prevents unsolicited transactions. However, malicious programs can still read all transactions displayed on a user PC. We prioritize preventing the compromise of long-term credentials over privacy concerns, but acknowledge that privacy of user transactions, as well as other account related information is very important. MP-Auth is designed primarily for financially-sensitive user authentication, and phishing attacks are also considered in that perspective. However, phishing attacks can target online services (e.g., social networking sites) which are not directly critical to a user's financial activities; information extracted from such attacks can be exploited for launching other attacks. Also, our proposed localized ID techniques only limit exploitation of long-term identity numbers. Designing techniques that may help improve privacy of other (breached) personal information remains as a larger and more elusive open problem.

In terms of protocol analysis, we analyze MP-Auth using a combination of BAN-like logic, an automated formal verification (software) tool called AVISPA, and a formal proof technique called PCL. Each of these techniques has its own advantages and limitations, and the security/crypto research community appears to be divided (see e.g., [143]) on how much confidence one may gain from each individual technique. We do not have any proof whether combining these three approaches offers any improved security guarantee, neither do we suggest any particular technique or a combination of techniques that may provide better security. Thus selecting proof techniques or confidence building steps for protocol analysis still remain an open problem.

We promote usability as a design goal throughout our proposals in this thesis. However, we do not conduct any formal user testing of our proposals. Carrying out such studies *properly* (e.g., through a combination of lab and field studies with a realistic user-base) appears to be non-trivial. We possibly cannot expect meaningful and consistent usability results from simple (e.g., lab-based) user studies; apparently many user studies in computer science test *learnability* instead of usability. (See Greenberg and Buxton [106] for a critique of the varying landscape of usability evaluation methods.) After learning something new, people's performance with respect to usability may change; for example, many people find the text-based editor `vi` usable, and many more cannot imagine editing documents without GUI-based `Microsoft Word`. On the positive side, current lab-based user studies may reveal obvious flaws in early prototypes. How to conduct user studies of new technologies that may provide better and more consistent results? We leave this as an open question. Perhaps, we need to ask ourselves whether user-testing is something better left for other experts (e.g., people from psychology).

Another important aspect related to our proposals that has not been discussed in detail is the economic cost of deployment. Even optimal or nearly optimal (from a security viewpoint) solutions may appear undesirable to vendors' financially-driven viewpoint. Security proponents often present *increased* customer confidence and trust in addition to reduced losses for adoption of security solutions by businesses. However, it is difficult to estimate any economic advantage from such increased consumer trust. Thus it remains an open question how to make an adequate business case for solutions that promise better security and privacy for consumers at the (initial) expense of vendors.

Bibliography

[1] M. Abadi, M. Burrows, C. Kaufman, and B. Lampson. Authentication and delegation with smart-cards. *Science of Computer Programming*, 21(2), 1993.

[2] ABC News. MySpace finds 29,000 sex offenders. News article (July 25, 2007). `http://www.abcnews.go.com/Technology/wireStory?id=3409947`.

[3] A. Acquisti, A. Friedman, and R. Telang. Is there a cost to privacy breaches? An event study. In *International Conference of Information Systems (ICIS'06)*, Milwaukee, WI, USA, Dec. 2006.

[4] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12), 1999.

[5] B. Adida. Beamauth: Two-factor web authentication with a bookmark. In *ACM Computer and Communications Security (CCS'07)*, Alexandria, VA, USA, 2007.

[6] S. Ahern, D. Eckles, N. Good, S. King, M. Naaman, and R. Nair. Over-exposed? Privacy patterns and considerations in online and mobile photo sharing. In *Conference on Human Factors in Computing Systems (CHI'07)*, San Jose, CA, USA, 2007.

[7] J. Aitel. The IPO of the 0day: Stock fluctuation from an unrecognized influence. In *Symposium on Security for Asia Network (SyScan'07)*, Singapore, July 2007. Keynote address.

[8] Algorithmic Research (ARX). PrivateServer Switch-HSM. White paper. `http://www.arx.com/documents/Switch-HSM.pdf`.

[9] R. Anderson. Why cryptosystems fail. *Communications of the ACM*, 37(11), Nov. 1994.

[10] R. Anderson, M. Bond, J. Clulow, and S. Skorobogatov. Cryptographic processors – a survey. *Proceedings of the IEEE*, 94(2), Feb. 2006. Invited paper.

[11] Anonymous. In the face of danger: Facial recognition and the limits of privacy law. *Harvard Law Review*, 120(7), May 2007.

[12] Anti-Phishing Working Group. Phishing activity trends report for April 2007. `http://www.antiphishing.org/reports/apwg_report_april_2007.pdf`.

[13] Anti-Phishing Working Group. Phishing Activity Trends Report, Q1/2008. `http://www.antiphishing.org/reports/apwg_report_Q1_2008.pdf`.

[14] Armando et al. The AVISPA tool for the automated validation of Internet security protocols and applications. In *Computer Aided Verification (CAV'05)*, volume 3576 of *LNCS*, 2005. Website: `http://www.avispa-project.org`.

[15] ArsTechnica.com. Half of Americans clueless about online threats. News article (Aug. 14, 2007).

[16] ArsTechnica.com. Yahoo Messenger and Windows Live Messenger get together. News Article (Sept. 27, 2006). `http://arstechnica.com/news.ars/post/20060927-7846.html`.

[17] P. Ashley, C. Powers, and M. Schunter. From privacy promises to privacy management: A new approach for enforcing privacy throughout an enterprise. In *New Security Paradigms Workshop (NSPW'02)*, Virginia Beach, VA, USA, Sept. 2002.

[18] J. Aycock and N. Friess. Spam zombies from outer space. In *EICAR Conference*, Hamburg, Germany, 2006.

[19] K. Bailey, A. Kapadia, L. Vongsathorn, and S. W. Smith. TwoKind authentication: Protecting private information in untrustworthy environments. In *ACM Workshop on Privacy in the Electronic Society (WPES'08)*, Alexandria, VA, USA, Oct. 2008.

[20] D. Balfanz and E. Felten. Hand-held computers can be better smart cards. In *USENIX Security Symposium*, Washington, DC, USA, Aug. 1999.

[21] S. B. Barnes. A privacy paradox: Social networking in the United States. *First Monday: Peer-reviewed Journal on the Internet*, 11(9), 2006.

[22] R. J. Bayardo and S. Thomschke. Exploiting the web for point-in-time file sharing (poster). In *World Wide Web Conference (WWW'05)*, 2005.

[23] R. J. Bayardo Jr., R. Agrawal, D. Gruhl, and A. Somani. YouServ: A web hosting and content sharing tool for the masses. In *World Wide Web Conference (WWW'02)*, 2002.

[24] BBC News. 'I was falsely branded a paedophile'. News article (Apr. 3, 2008). `http://news.bbc.co.uk/1/hi/magazine/7326736.stm`.

[25] BBC News. Malware 'hijacks Windows Updates'. News article (May 16, 2007).

[26] BBC News. Monster attack steals user data. News article (Aug. 21, 2007). `http://news.bbc.co.uk/2/hi/technology/6956349.stm`.

[27] M. Beaumont-Gay, K. Eustice, and P. Reiher. Information protection via environmental data tethers. In *New Security Paradigms Workshop (NSPW'07)*, New Hampshire, USA, Sept. 2007.

[28] M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *AsiaCrypt'00*, Kyoto, Japan, Dec. 2000.

[29] M. Bellare and P. Rogaway. Entity authentication and key distribution. In *Advances in Cryptology (Crypto'93)*, Santa Barbara, CA, USA, Aug. 1993.

[30] A. Bellissimo, J. Burgess, and K. Fu. Secure software updates: Disappointments and new challenges. In *USENIX Workshop on Hot Topics in Security (HotSec'06)*, Vancouver, BC, Canada, July 2006.

[31] J. Benamati, M. A. Serva, and M. A. Fuller. Are trust and distrust distinct constructs? An empirical study of the effects of trust and distrust among online banking users. In *IEEE Hawaii International Conference on System Sciences (HICCS'06)*, Jan. 2006.

[32] O. Berkman and O. M. Ostrovsky. The unbearable lightness of PIN cracking. In *Financial Cryptography and Data Security (FC'07)*, Scarborough, Trinidad and Tobago, Feb. 2007.

[33] Beskerming.com. How the online trust model is broken - the BankOfIndia.com attack. News article (Aug. 31, 2007).

[34] A. Biryukov, J. Lano, and B. Preneel. Cryptanalysis of the alleged SecurID hash function. In *Selected Areas in Cryptography (SAC'03)*, volume 3006 of *LNCS*, Ottawa, Canada, Aug. 2003.

[35] M. Bishop. Psychological acceptability revisited. In "Security and Usability: Designing Secure Systems that People Can Use." Edited by L. Cranor and S. Garfinkel. O'Reilly, 2005.

[36] J. Black and P. Rogaway. Ciphers with arbitrary finite domains. In *RSA Security (CT-RSA)*, San Jose, CA, USA, Feb. 2002.

[37] J. Blascovich. Mind games: A psychological analysis of common email scams. McAfee Avert Labs white paper (June 25, 2007). `http://www.mcafee.com/us/local_content/white_papers/wp_mind_games_en.pdf`.

[38] M. Bond. Phantom withdrawals: On-line resources for victims of ATM fraud. `http://www.phantomwithdrawals.com`.

[39] M. Bond. Understanding security APIs. Ph.D. Thesis, Computer Laboratory, University of Cambridge, 2004.

[40] M. Bond. Attacks on cryptoprocessor transaction sets. In *Workshop on Cryptographic Hardware and Embedded Systems (CHES'01)*, Paris, France, May 2001.

[41] M. Bond and P. Zielinski. Decimalisation table attacks for PIN cracking. Technical report (UCAM-CL-TR-560), Computer Laboratory, University of Cambridge, 2003.

[42] M. Bond and P. Zielinski. Encrypted? Randomised? Compromised? (When cryptographically secured data is not secure). In *Workshop on Cryptographic Algorithms and their Uses*, Gold Coast, Australia, July 2004.

[43] N. Borisov, I. Goldberg, and E. Brewer. Off-the-record communication, or, why not to use PGP. In *ACM Workshop on Privacy in the Electronic Society (WPES'04)*, Washington, DC, USA, Oct. 2004.

[44] C. Boyd and A. Mathuria. *Protocols for Authentication and Key Establishment*. Information Security and Cryptography Book Series (editors: Ueli Maurer and Ronald L. Rivest). Springer, 2003.

[45] BusinessWeek. Social-networking sites a 'hotbed' for spyware. News article (Aug. 18, 2006). `http://www.msnbc.msn.com/default.aspx/id/14413906/`.

[46] CA Virus Information Center. Win32.Grams.I, Feb. 2005. `http://www3.ca.com`.

[47] K. Cameron. The laws of identity. Blog article (May 12, 2005). `http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf`.

[48] CBC.ca. 4 charged after school protest over Facebook suspensions. News article (Mar. 23, 2007). `http://www.cbc.ca/canada/toronto/story/2007/03/23/protest-birchmount.html`.

[49] ChannelRegister.co.uk. IT pro admits stealing 8.4M consumer records. News article (Dec. 4, 2007). `http://www.channelregister.co.uk/2007/12/04/admin_steals_consumer_records/`.

[50] W. Cheswick. Johnny can obfuscate; beyond mothers maiden name. In *USENIX Workshop on Hot Topics in Security (HotSec'06)*, Vancouver, BC, Canada, July 2006.

[51] S. Chiasson, P. van Oorschot, and R. Biddle. A usability study and critique of two password managers. In *USENIX Security Symposium*, Vancouver, Canada, Aug. 2006.

[52] W. Chung and J. Paynter. An evaluation of Internet banking in New Zealand. In *IEEE Hawaii International Conference on System Sciences (HICCS'02)*, Jan. 2002.

[53] D. E. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. van Dijk, S. Devadas, and R. L. Rivest. The untrusted computer problem and camera-based authentication. In *Pervasive Computing*, volume 2414 of *LNCS*, Zurich, Switzerland, Aug. 2002.

[54] J. Clulow. The design and analysis of cryptographic APIs for security devices. Masters Thesis, University of Natal, Durban, South Africa, 2003.

[55] CNN.com. FBI seeks stolen personal data on 26 million vets. News article (May 23, 2006). `http://www.cnn.com/2006/US/05/22/vets.data/index.html`.

[56] Commtouch.com. Malware outbreak trend report: Storm-Worm. Online article (Jan. 31, 2007). `http://www.commtouch.com/downloads/Storm-Worm_MOTR.pdf`.

[57] Computerworld.com. Malware count blows past 1M mark. News article (Apr. 8, 2008).

[58] ComputerWorld.com. Scope of TJX data breach doubles: 94M cards now said to be affected. News article (Oct. 24, 2007).

[59] ComputerWorld.com. Symantec false positive cripples thousands of Chinese PCs. News article (May 18, 2007).

[60] ComputerWorld.com. TJX violated nine of 12 PCI controls at time of breach, court filings say. News article (Oct. 26, 2007).

[61] Consumeraffairs.com. Consumers losing confidence in online commerce, banking. News article (June 28, '05).

[62] Cronto.com. Visual cryptogram. `http://www.cronto.com/visual_cryptogram.htm`.

[63] DarkReading.com. Antivirus tools underperform when tested in LinuxWorld 'Fight Club'. News article (Aug. 9, 2007).

[64] A. Datta, A. Derek, J. C. Mitchell, and A. Roy. Protocol composition logic (PCL). *Electronic Notes in Theoretical Computer Science (ENTCS)*, 172, Apr. 2007.

[65] D. Davis. Compliance defects in public-key cryptography. In *USENIX Security Symposium*, San Jose, CA, USA, July 1996.

[66] F. Dawson and T. Howes. vCard MIME directory profile, 1998. RFC 2426, Status: Standards Track.

[67] R. Dhamija, J. Tygar, and M. Hearst. Why phishing works. In *Conference on Human Factors in Computing Systems (CHI'06)*, Montreal, QC, Canada, Apr. 2006.

[68] W. Diffie, P. van Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2(2), 1992.

[69] D. Dolev and A. C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2), Mar. 1983.

[70] J. S. Downs, M. Holbrook, and L. F. Cranor. Decision strategies and susceptibility to phishing. In *Symposium On Usable Privacy and Security (SOUPS'06)*, Pittsburgh, PA, USA, July 2006.

[71] S. Drimer, S. J. Murdoch, and R. Anderson. Thinking inside the box: System-level failures of tamper proofing. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2008.

[72] S. Drimer, S. J. Murdoch, and R. Anderson. Optimised to fail: Card readers for online banking. In *Financial Cryptography and Data Security (FC'09)*, Barbados, Feb. 2009.

[73] C. Dwyer, S. Hiltz, and K. Passerini. Trust and privacy concern within social networking sites: A comparison of Facebook and MySpace. In *Americas Conference on Information Systems (AMCIS'07)*, Keystone, Colorado, USA, Aug. 2007.

[74] K. Edge, R. Raines, M. Grimaila, R. Baldwin, R. Bennington, and C. Reuter. The use of attack and protection trees to analyze security for an online banking system. In *IEEE Hawaii International Conference on System Sciences (HICCS'07)*, Jan. 2007.

[75] Electronic Privacy Information Center (EPIC). Protecting the privacy of the Social Security Number from identity theft. EPIC testimony before the Committee on Ways and Means in the U.S. House of Representatives (June 21, 2007). `http://www.epic.org/privacy/ssn/idtheft_test_062107.pdf`.

[76] C. M. Ellison, C. Hall, R. Milbert, and B. Schneier. Protecting secret keys with personal entropy. *Future Generation Computer Systems*, 16(4), Feb. 2000.

[77] Entrust.com. Katrina scams show browser security model is broken. Entrust blog (Sept. 9, 2005).

[78] K. Erickson and P. N. Howard. A case of mistaken identity? News accounts of hacker, consumer, and organizational responsibility for compromised digital records. *Journal of Computer-Mediated Communication*, 12(4), July 2007.

[79] eWeek.com. Microsoft patches causing breakages, lockups. News article (Apr. 17, 2006).

[80] eWeek.com. Microsoft says recovery from malware becoming impossible. News article (Apr. 4, 2006).

[81] F-Secure. F-Secure virus descriptions: Cabir, June 2004.

[82] F-Secure. F-Secure trojan information pages: Redbrowser.A, Mar. 2006.

[83] Federal Financial Institutions Examination Council (FFIEC). FFIEC guidance: Authentication in an Internet banking environment, Oct. 2005. `http://www.fdic.gov/news/news/financial/2005/fil10305.html`.

[84] Federal Trade Commission. Identity theft survey report, Sept. 2003. `http://www.ftc.gov/os/2003/09/synovatereport.pdf`.

[85] Federal Trade Commission. Security in numbers: SSNs and ID theft workshop, Dec. 2007. `http://ftc.gov/bcp/workshops/ssn/index.shtml`.

[86] R. Feizy. An evaluation of identity on online social networking: MySpace (poster). In *ACM Hypertext and Hypermedia (Hypertext'07)*, Manchester, UK, Sept. 2007.

[87] E. W. Felten, D. Balfanz, D. Dean, and D. S. Wallach. Web spoofing: An Internet con game. In *National Information Systems Security Conference*, Oct. 1997.

[88] Finextra.com. ABN Amro compensates victims of 'man-in-the-middle' attack. News article (Apr. 2, 2007). `http://www.finextra.com/fullstory.asp?id=16750`.

[89] Finjan Malicious Code Research Center. Web security trends report – Q3/2007. `http://www.finjan.com/GetObject.aspx?ObjId=506`.

[90] D. Florêncio and C. Herley. A large-scale study of web password habits. In *World Wide Web Conference (WWW'07)*, Banff, Alberta, Canada, May 2007.

[91] D. Florêncio, C. Herley, and B. Coskun. Do strong web passwords accomplish anything? In *USENIX Workshop on Hot Topics in Security (HotSec'07)*, Boston, MA, USA, Aug. 2007.

[92] A. Forget, S. Chiasson, P. van Oorschot, and R. Biddle. Improving text passwords through persuasion. In *Symposium on Usable Privacy and Security (SOUPS'08)*, Pittsburgh, PA, USA, July 2008.

[93] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. HTTP authentication: Basic and digest access authentication, June 1999. RFC 2617, Status: Standards Track.

[94] S. Furnell. Authenticating ourselves: will we ever escape the password? *Network Security*, 2005(3):8–13, Mar. 2005.

[95] C. Gates and J. Slonim. Owner-controlled information. In *New Security Paradigms Workshop (NSPW'03)*, Ascona, Switzerland, Aug. 2003.

[96] S. Gaw and E. W. Felten. Password management strategies for online accounts. In *Symposium On Usable Privacy and Security (SOUPS'06)*, Pittsburgh, PA, USA, July 2006.

[97] M. Geist. Facing up to Facebook fears. BBC news article (May 9, 2007). `http://news.bbc.co.uk/2/hi/technology/6639417.stm`.

[98] R. L. Glass. Patching is alive and, lamentably, thriving in the real-time world. *ACM SIGPLAN Notices*, 13(3), 1978.

[99] V. D. Gligor and P. Donescu. Fast encryption and authentication: XCBC encryption and XECB authentication modes. In *Workshop on Fast Software Encryption (FSE'01)*, Yokohama, Japan, Apr. 2001.

[100] Globe and Mail. globeandmail.com: Mary Kirwan. News article (Nov. 16, 2006). `http://www.theglobeandmail.com/servlet/story/RTGAM.20061116.gtkirwan1116/BNStory/Technology/home`.

[101] A. Gostev and A. Shevchenko. Kaspersky security bulletin, January - June 2006: Malicious programs for mobile devices, Sept. 2006. `http://www.viruslist.com`.

[102] Government Accountability Office (GAO). Personal information: Data breaches are frequent, but evidence of resulting identity theft is limited; however, the full extent is unknown, June 2007. Report to Congressional Requesters, GAO-07-737, `http://www.gao.gov/new.items/d07935t.pdf`.

[103] Government Accountability Office (GAO). Social Security Numbers: Use is widespread and protection could be improved, June 2007. Testimony Before the Subcommittee on Social Security, Committee on Ways and Means, House of Representatives, GAO-07-1023T, `http://www.gao.gov/new.items/d071023t.pdf`.

[104] Government Reform Committee. Agency data breaches since January 1, 2003. Staff Report, U.S. House of Representatives (Oct. 13, 2006). `http://oversight.house.gov/documents/20061013145352-82231.pdf`.

[105] R. Graubart. On the need for a third form of access control. In *12th National Computer Security Conference*, Baltimore, MD, USA, Oct. 1989.

[106] S. Greenberg and B. Buxton. Usability evaluation considered harmful (some of the time). In *Conference on Human Factors in Computing Systems (CHI'08)*, Florence, Italy, Apr. 2008.

[107] P. Greenspun. Mobile phone as home computer, Sept. 2005. `http://philip.greenspun.com/business/mobile-phone-as-home-computer`.

[108] S. J. Greenwald, K. G. Olthoff, V. Raskin, and W. Ruch. The user non-acceptance paradigm: INFOSEC's dirty little secret (panel discussion). In *New Security Paradigms Workshop (NSPW'04)*, Nova Scotia, Canada, Sept. 2004.

[109] V. Griffith and M. Jakobsson. Messin' with Texas, deriving mother's maiden names using public records. In *Applied Cryptography and Network Security (ACNS'05)*, New York, NY, USA, June 2005.

[110] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *ACM Workshop on Privacy in the Electronic Society (WPES'05)*, Alexandria, VA, USA, Nov. 2005.

[111] J. Grossklags and N. Good. Empirical studies on software notices to inform policy makers and usability designers. In *Workshop on Usable Security (USEC'07)*, Lowlands, Scarborough, Trinidad and Tobago, Feb. 2007.

[112] Guardian.co.uk. Lost in the post - 25 million at risk after data discs go missing. News article (Nov. 21, 2007). `http://www.guardian.co.uk/politics/2007/nov/21/immigrationpolicy.economy3`.

[113] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, San Jose, CA, USA, 2008.

[114] S. Halevi and H. Krawczyk. Public-key cryptography and password protocols. *ACM Transactions on Information and Systems Security (TISSEC)*, 2(3), Aug. 1999.

[115] N. Haller. The S/KEY one-time password system. In *Symposium on Network and Distributed System Security (NDSS'94)*, San Diego, CA, USA, Feb. 1994.

[116] R. Hasan and W. Yurcik. A statistical analysis of disclosed storage security breaches. In *ACM Workshop on Storage Security and Survivability (StorageSS'06)*, Alexandria, VA, USA, Oct. 2006.

[117] C. He, M. Sundararajan, A. Datta, A. Derek, and J. C. Mitchell. A modular correctness proof of IEEE 802.11i and TLS. In *ACM Computer and Communications Security (CCS'05)*, Alexandria, VA, USA, Nov. 2005.

[118] J. Heasman. Implementing and detecting a PCI rootkit. White paper (Nov. 15, 2006). `http://www.ngssoftware.com`.

[119] G. Heiser. Secure embedded systems need microkernels. ;login:, Dec. 2005.

[120] Help Net Security (HNS). Server with top-secret data stolen from Forensic Telecommunications Services. News article (Aug. 13, 2007). `http://www.net-security.org/secworld.php?id=5418`.

[121] M. Hertzum, N. Jørgense, and M. Nørgaar. Usable security and e-banking: Ease of use vis-à-vis security. *Australasian Journal of Information Systems*, 11, 2004.

[122] A. Herzogl and N. Shahmehri. Usability and security of personal firewalls. In *IFIP Information Security Conference (SEC'07)*, Sandton, South Africa, May 2007.

[123] C. J. Hoofnagle. How SSNs are used to commit ID theft: Synthetic identity theft. In *Security in Numbers: SSNs and ID Theft Workshop*, Dec. 2007. Panel presentation at the workshop, hosted by the FTC.

[124] ICANN Security and Stability Advisory Committee. Domain name hijacking: Incidents, threats, risks, and remedial actions, July 2005. `http://www.icann.org`.

[125] Identity Theft Resource Center (ITRC). ITRC 2008 breach list. Security breaches from 2005 to 2008 (Apr. 8, 2008). `http://www.idtheftcenter.org/artman2/publish/lib_survey/ITRC_2008_Breach_List.shtml`.

[126] InformationWeek.com. Storm worm botnet more powerful than top supercomputers. News article (Sept. 6, 2007).

[127] International Organization for Standardization (ISO). Banking – Personal Identification Number (PIN) management and security – Part 1: Basic principles and requirements for online PIN handling in ATM and POS systems, Apr. 2002. International Standard, ISO 9564-1.

[128] jabberd project. jabberd2 XMPP server. Version 2.1.6. `http://jabberd.jabberstudio.org/2/`.

[129] C. Jackson, D. Boneh, and J. Mitchell. Spyware resistant web authentication using virtual machines. Technical report (2006). `http://crypto.stanford.edu/spyblock`.

[130] C. Jackson, D. Boneh, and J. Mitchell. Transaction generators: Root kits for web. In *USENIX Workshop on Hot Topics in Security (HotSec'07)*, Boston, MA, USA, Aug. 2007.

[131] C. Jackson, D. Simon, D. Tan, and A. Barth. An evaluation of Extended Validation and picture-in-picture phishing attacks. In *Workshop on Usable Security (USEC'07)*, Lowlands, Scarborough, Trinidad and Tobago, Feb. 2007.

[132] T. Jagatic, N. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10), Oct. 2007.

[133] R. C. Jammalamadaka, T. van der Horst, S. Mehrotra, K. Seamons, and N. Venkatasuramanian. Delegate: A proxy based architecture for secure website access from an untrusted machine. In *Annual Computer Security Applications Conference (ACSAC'06)*, Miami Beach, FL, USA, Dec. 2006.

[134] N. Jin and M. Fei-Cheng. Network security risks in online banking. In *IEEE Wireless Communications, Networking and Mobile Computing (WiMob'05)*, Montreal, QC, Canada, Aug. 2005.

[135] M. E. Johnson and S. Dynes. Inadvertent disclosure – information leaks in the extended enterprise. In *Workshop on the Economics of Information Security (WEIS'07)*, Pittsburgh, PA, USA, June 2007.

[136] M. Just. Designing secure yet usable challenge question authentication systems. In "Security and Usability: Designing Secure Systems that People Can Use." Edited by L. Cranor and S. Garfinkel. O'Reilly, 2005.

[137] D. Kaminsky. Black Ops 2008 – It's the end of the cache as we know it. In *Black Hat USA*, Las Vegas, NV, USA, Aug. 2008.

[138] H. Karjaluoto, T. Koivumäki, and J. Salo. Individual differences in private banking: Empirical evidence from Finland. In *IEEE Hawaii International Conference on System Sciences (HICCS'03)*, Jan. 2003.

[139] K. Karvonen. The beauty of simplicity. In *ACM Conference on Universal Usability*, Arlington, VA, USA, Nov. 2000.

[140] Kaspersky.com. Malicious mass mailing sent using McAfee email address. Virus News (Nov. 2, 2006).

[141] Keynote.com. Online banking critical to bank selection and brand perception. Press release (Jan. 6, 2005).

[142] S. T. King, P. M. Chen, Y.-M. Wang, C. Verbowski, H. J. Wang, and J. R. Lorch. SubVirt: Implementing malware with virtual machines. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2006.

[143] N. Koblitz and A. J. Menezes. Another look at “provable security”. *Journal of Cryptology*, 20(1), 2007.

[144] D. Kuhlman, R. Moriarty, T. Braskich, S. Emeott, and M. Tripunitara. A correctness proof of a mesh security architecture. In *IEEE Computer Security Foundations Symposium (CSF'08)*, Pittsburgh, PA, USA, June 2008.

[145] C. Kuo, S. Romanosky, and L. F. Cranor. Human selection of mnemonic phrase-based passwords. In *Symposium On Usable Privacy and Security (SOUPS'06)*, Pittsburgh, PA, USA, July 2006.

[146] K. Kursawe and S. Katzenbeisser. Computing under occupation. In *New Security Paradigms Workshop (NSPW'07)*, New Hampshire, USA, Sept. 2007.

[147] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11), Nov. 1981.

[148] B. Laurie and A. Singer. Choose the red pill and the blue pill. In *New Security Paradigms Workshop (NSPW'08)*, Lake Tahoe, CA, USA, Sept. 2008.

[149] Y. Li and X. Zhang. A security-enhanced one-time payment scheme for credit card. In *IEEE Workshop on Research Issues on Data Engineering: Web Services for E-Commerce and E-Government Applications (RIDE'04)*, Boston, MA, USA, Mar. 2004.

[150] Y. Li and X. Zhang. Securing credit card transactions with one-time payment scheme. *Journal of Electronic Commerce Research and Applications*, 4(4), 2005.

[151] Liberty Alliance. Liberty ID-WSF People Service – federated social identity. White paper (Dec. 5, 2005). `http://www.projectliberty.org`.

[152] MacDevCenter.com. How Paris got hacked? News article (Feb. 22, 2005), `http://www.macdevcenter.com/pub/a/mac/2005/01/01/paris.html`.

[153] M. Mannan. AVISPA test code for Mobile Password Authentication (MP-Auth). `http://www.scs.carleton.ca/~mmannan/mpauth`.

[154] M. Mannan and P. van Oorschot. Secure public instant messaging: A survey. In *Privacy, Security and Trust (PST'04)*, Fredericton, NB, Canada, Oct. 2004.

[155] M. Mannan and P. van Oorschot. A protocol for secure public instant messaging. In *Financial Cryptography and Data Security (FC'06)*, Anguilla, British West Indies, 2006.

[156] N. B. Margolin, M. K. Wright, and B. N. Levine. Guardian: A framework for privacy control in untrusted environments, June 2004. Technical Report 04-37 (University of Massachusetts, Amherst).

[157] V. Mayer-Schönberger. Useful void: The art of forgetting in the age of ubiquitous computing. Harvard KSG Faculty Research Working Paper Series, article number RWP07-022, Apr. 2007.

[158] McAfee and National Cyber Security Alliance (NCSA). McAfee-NCSA online safety study, Oct. 2007.

[159] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2005.

[160] J. M. McCune, A. Perrig, and M. K. Reiter. Bump in the Ether: A framework for securing sensitive user input. In *USENIX Annual Technical Conference*, Boston, MA, USA, 2006.

[161] D. McDonald. A convention for human-readable 128-bit keys, Dec. 1994. RFC 1751 (Informational). `http://www.ietf.org/rfc/rfc1751.txt`.

[162] Microsoft. Password checker. `http://www.microsoft.com/athome/security/privacy/password_checker.mspx`.

[163] Microsoft. Strong passwords: How to create and use them. Online article (Mar. 22, 2006). `http://www.microsoft.com/protect/yourself/password/create.mspx`.

[164] Microsoft Support. Detailed installation walkthrough for Windows XP Service Pack 2. `http://support.microsoft.com`.

[165] J. Milletary. Technical trends in phishing attacks. US-CERT, Reading room article, `http://www.us-cert.gov/reading_room/phishing_trends0511.pdf`.

[166] Mobile Antivirus Researchers Association. Analyzing the crossover virus: The first PC to Windows handheld cross-infector, 2006. `http://www.informit.com`.

[167] Mobile electronic Transactions (MeT) Ltd. Personal Transaction Protocol Version 1.0 (Draft Specification), Jan. 2002. `http://www.mobiletransaction.org/pdf/R200/specifications/MeT_PTP_v100.pdf`.

[168] Mobile Phone Work Group. TCG mobile trusted module specification. Specification version 1.0, Revision 1, 12 June 2007.

[169] I. Molloy, J. Li, and N. Li. Dynamic virtual credit card numbers. In *Financial Cryptography and Data Security (FC'07)*, Scarborough, Trinidad and Tobago, Feb. 2007.

[170] R. Morris and K. Thompson. Password security: A case history. *Communications of the ACM*, 22(11), Nov. 1979.

[171] A. Moshchuk, T. Bragin, S. D. Gribble, and H. Levy. A crawler-based study of spyware in the web. In *Network and Distributed System Security (NDSS'06)*, San Diego, CA, USA, Feb. 2006.

[172] MSNBC. Breach exposes 4.2 million credit, debit cards. News article (Mar. 17, 2008). `http://www.msnbc.msn.com/id/23678909/`.

[173] Mudge and Kingpin. Initial cryptanalysis of the RSA SecurID algorithm, 2001. White paper. `http://www.linuxsecurity.com/resource_files/cryptography/initial_securid_analysis.pdf`.

[174] NACE Spotlight Online. The issues surrounding college recruiting and social networking web sites. News article (June 22, 2006). `http://career.studentaffairs.duke.edu/undergrad/find_job/consider/nace_socialnetworks.html`.

[175] D. Nali and P. van Oorschot. Videoticket: Detecting identity fraud attempts via audiovisual certificates and signatures. In *New Security Paradigms Workshop (NSPW'07)*, New Hampshire, USA, Sept. 2007.

[176] D. Nali and P. van Oorschot. CROO: A universal infrastructure and protocol to detect identity fraud. In *European Symposium on Research in Computer Security (ESORICS'08)*, Malaga, Spain, Oct. 2008.

[177] National Cyber Security Alliance. CA/NCSA social networking cyber security survey. Online article (Sept. 2006). `http://staysafeonline.org/features/SocialNetworkingReport.ppt`.

[178] National Post. Watchdog pushed CIBC on lost file. News article (Jan. 26, 2007). `http://www.canada.com`.

[179] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12), 1978.

[180] Netcraft. Fraudsters attack two-factor authentication. News article (July 13, 2006). `http://news.netcraft.com/archives/2006/07/13/fraudsters_attack_twofactor_authentication.html`.

[181] Netcraft. More than 450 phishing attacks used SSL in 2005, Dec. 2005. `http://news.netcraft.com`.

[182] Netcraft.com. Bank, customers spar over phishing losses. News article (Sept. 13, 2006).

[183] Netcraft.com. MySpace accounts compromised by phishers. News article (Oct. 27, 2006). `http://news.netcraft.com/archives/2006/10/27/myspace_accounts_compromised_by_phishers.html`.

[184] B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications*, 32(9), Sept. 1994.

[185] New York Times. For some, online persona undermines a résumé. News article (June 11, 2006). http://www.nytimes.com/2006/06/11/us/11recruit.html.

[186] New York Times. How to lose your job on your own time. News article (Dec. 30, 2007). http://www.nytimes.com/2007/12/30/business/30digi.html.

[187] New Zealand Bankers' Association (NZBA). Code of banking practice. Fourth Edition (July, 2007).

[188] M. Nilsson, A. Adams, and S. Herd. Building security and trust in online banking (extended abstract). In *Conference on Human Factors in Computing Systems (CHI'05)*, Portland, OR, USA, Apr. 2005.

[189] C. Nodder. Users and trust: A Microsoft case study. In "Security and Usability: Designing Secure Systems that People Can Use." Edited by L. Cranor and S. Garfinkel. O'Reilly, 2005.

[190] B. O'Connor. Greater than 1: Defeating "strong" authentication in web applications. In *Defcon 15*, Las Vegas, NV, USA, Aug. 2007.

[191] Office of the Privacy Commissioner of Canada. Guidelines for identification and authentication, Oct. 2006. http://www.privcom.gc.ca/information/guide/auth_061013_e.asp.

[192] OpenWall.com. John the Ripper password cracker. http://www.openwall.com/john/.

[193] A. Oprea, D. Balfanz, G. Durfee, and D. Smetters. Securing a remote terminal application with a mobile trusted device. In *Annual Computer Security Applications Conference (ACSAC'04)*, Tucson, AZ, USA, Dec. 2004.

[194] O. M. Ostrovsky. Vulnerabilities in the financial PIN processing API. Masters Thesis, Tel Aviv University, 2006.

[195] B. Parno, C. Kuo, and A. Perrig. Phoolproof phishing prevention. In *Financial Cryptography and Data Security (FC'06)*, volume 4107 of *LNCS*, Anguilla, British West Indies, Feb. 2006.

[196] A. S. Patrick. Monitoring corporate password sharing using social network analysis. In *International Sunbelt Social Network Conference*, St. Pete Beach, Florida, USA, Jan. 2008.

[197] A. M. Payton. Data security breach: Seeking a prescription for adequate remedy. In *ACM Conference on Information Security Curriculum Development (InfoSecCD'06)*, Kennesaw, GA, USA, Sept. 2006.

[198] PeiterZ@silence.secnet.com. Weaknesses in SecurID. White paper. http://www.tux.org/pub/security/secnet/papers/secureid.pdf.

[199] A. Perrig and D. Song. Hash visualization: A new technique to improve real-world security. In *Cryptographic Techniques and E-Commerce (CrypTEC'99)*, Hong Kong, July 1999.

[200] Pidgin project. Pidgin: A multi-protocol IM client. Version 2.0.1. http://www.pidgin.im/.

[201] Ponemon Institute. 2007 annual study: U.S. cost of a data breach, Nov. 2007. Research report sponsored by PGP and Symantec, http://www.pgp.com/downloads/research_reports/.

[202] PrisonPlanet.com. The Facebook.com: Big brother with a smile. News article (June 9, 2005). http://www.prisonplanet.com/articles/june2005/090605thefacebook.htm.

[203] Privacy Rights Clearing House. A chronology of data breaches. http://www.privacyrights.org/ar/ChronDataBreaches.htm.

[204] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iFRAMEs point to us. In *USENIX Security Symposium*, San Jose, CA, USA, July 2008.

[205] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser: Analysis of web-based malware. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, Cambridge, MA, USA, Apr. 2007.

[206] A. Rabkin. Personal knowledge questions for fallback authentication. In *Symposium on Usable Privacy and Security (SOUPS'08)*, Pittsburgh, PA, USA, July 2008.

[207] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. My botnet is bigger than yours (maybe, better than yours): Why size estimates remain challenging. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots'07)*, Cambridge, MA, USA, Apr. 2007.

[208] D. Ramsbrock, R. Berthier, and M. Cukier. Profiling attacker behavior following SSH compromises. In *IEEE/IFIP Dependable Systems and Networks (DSN'07)*, Edinburgh, UK, June 2007.

[209] D. Rand. Threats when using online social networks. CSIS Security Group (a Danish IT security company; article published on May 16, 2007). http://www.csis.dk/dk/forside/LinkedIn.pdf.

[210] Redmondmag.com. Coreflood trojan stole 500G of personal financial data. News article (Aug. 7, 2008). http://redmondmag.com/news/article.asp?editorialsid=10111.

[211] Reuters UK. Networking sites a goldmine for ID fraudsters. News article (July 19, 2007). http://uk.reuters.com/article/personalFinanceNews/idUKHIL95513120070719.

[212] S. Romanosky, R. Telang, and A. Acquisti. Do data breach disclosure laws reduce identity theft? In *Workshop on the Economics of Information Security (WEIS'08)*, Hanover, NH, USA, June 2008.

[213] D. Rosenblum. What anyone can know: The privacy risks of social networking sites. *IEEE Security and Privacy*, 5(3), May 2007.

[214] B. Ross, C. Jackson, N. Miyake, D. Boneh, and J. C. Mitchell. Stronger password authentication using browser extensions. In *USENIX Security Symposium*, Baltimore, MD, USA, 2005.

[215] V. Roth, K. Richter, and R. Freidinger. A PIN-entry method resilient against shoulder surfing. In *ACM Computer and communications Security (CCS'04)*, Washington, DC, USA, Oct. 2004.

[216] A. Roy, A. Datta, A. Derek, J. C. Mitchell, and J.-P. Seifert. *Secrecy Analysis in Protocol Composition Logic.* Formal Logical Methods for System Security and Correctness, IOS Press, 2008. Volume based on presentations at Summer School Marktoberdorf, Germany, 2007.

[217] A. Rubin. Independent one-time passwords. *USENIX Journal of Computing Systems*, 9(1), Feb. 1996.

[218] A. Rubin and R. Wright. Off-line generation of limited-use credit card numbers. In *Financial Cryptography (FC'01)*, Grand Cayman, British West Indies, Feb. 2001.

[219] J. Rutkowska. Introducing Blue Pill, 2006. Presented at SyScan, http://theinvisiblethings.blogspot.com/2006/06/introducing-blue-pill.html.

[220] P. Saint-Andre. Extensible messaging and presence protocol (XMPP): Core, Oct. 2004. RFC 3920, Status: Standards Track.

[221] P. Saint-Andre. Extensible messaging and presence protocol (XMPP): Instant messaging and presence, 2004. RFC 3921, Status: Standards Track.

[222] P. Saint-Andre. Internationalized resource identifiers (IRIs) and uniform resource identifiers (URIs) for the extensible messaging and presence protocol (XMPP), July 2006. RFC 4622, Status: Standards Track.

[223] Samuelson Law, Technology & Public Policy Clinic, University of California-Berkeley School of Law. Security breach notification laws: Views from chief security officers, Dec. 2007. http://groups.ischool.berkeley.edu/samuelsonclinic/files/cso_study.pdf.

[224] SANS Institute Internet Storm Center. Windows XP: Surviving the first day, Nov. 2003.

[225] SANS Internet Storm Center. Fake microsoft patch email -> fake spyware doctor! Handler's diary (June 26, 2007).

[226] SANS Internet Storm Center. MySpace phish and drive-by attack vector propagating Fast Flux network growth. SANS handler's diary (June 26, 2007). http://isc.sans.org/diary.html?storyid=3060.

[227] SANS Internet Storm Center. Symantec false-positive on Filezilla, NASA World Wind. Handler's diary (July 16, 2007).

[228] S. Saroiu and A. Wolman. SpySaver: Using incentives to address spyware. In *ACM Workshop on the Economics of Networks, Systems, and Computation (NetEcon'08)*, Seattle, WA, USA, Aug. 2008.

[229] M. A. Sasse, S. Brostoff, and D. Weirich. Transforming the 'weakest link' - a human/computer interaction approach to usable and effective security. *BT Technology*, 19(3), 2001.

[230] M. A. Sasse and I. Flechais. Usable security: Why do we need it? how do we get it? In "Security and Usability: Designing Secure Systems that People Can Use." Edited by L. Cranor and S. Garfinkel. O'Reilly, 2005.

[231] scanit.be. Browser security test: A year of bugs, 2004. http://bcheck.scanit.be.

[232] B. Schneier. The curse of the secret question. Blog post (Feb. 11, 2005), http://www.schneier.com/blog/archives/2005/02/the_curse_of_th.html.

[233] B. Schneier. Real-world passwords. Analysis of 34,000 MySpace.com userid-password pairs. Blog post (Dec. 14, 2006). http://www.schneier.com/blog/archives/2006/12/realworld_passw.html.

[234] B. Schneier. Two-factor authentication: Too little, too late. *Communications of the ACM*, 48(4):136, 2005.

[235] Secunia. 1.91% of all PCs are fully patched! Secunia blog (Dec. 3, 2008). http://secunia.com/blog/37.

[236] SecurityFocus.com. Bot spreads through antivirus, Windows flaws. News article (Nov. 28, 2006).

[237] SecurityFocus.com. Employee steals 2.3m records from data firm. News article (July 5, 2007). http://www.securityfocus.com/brief/541.

[238] SecurityFocus.com. Image attack on MySpace boosts phishing exposure. News article (June 11, 2007). http://www.securityfocus.com/brief/522.

[239] SecurityFocus.com. QuickTime worm uses MySpace to spread. News article (Apr. 12, 2006). http://www.securityfocus.com/brief/375.

[240] M. Shackman. Platform security - a technical overview, Nov. 2006. Symbian Developer Network article. http://developer.symbian.com/main/oslibrary/symbian_os_papers/miscellaneous.jsp.

[241] A. Shamir. SecureClick: A web payment system with disposable credit card numbers. In *Financial Cryptography (FC'01)*, Grand Cayman, British West Indies, Feb. 2001.

[242] A. Shipp. Targeted trojan attacks and industrial espionage. In *Virus Bulletin Conference (VB'06)*, Montreal, Canada, Oct. 2006.

[243] Silicon.com Staff. Banks must boost security to drive online banking, Mar. 2005. Forrester Research report. http://software.silicon.com.

[244] A. Singer. Life without firewalls. *;login: The USENIX Magazine*, 28(6), 2003.

[245] S. Singh. The social dimensions of the security of Internet banking. *Journal of Theoretical and Applied Electronic Commerce Research*, 1(2), 2006.

[246] R. E. Smith. The strong password dilemma. Chapter 6 in "Authentication: From Passwords to Public Keys", Addison-Wesley, 2002. Excerpt available at http://www.cryptosmith.com/sanity/pwdilemma.html.

[247] D. J. Solove. Identity theft, privacy, and the architecture of vulnerability. *Hastings Law Journal*, 54, 2003. Available at SSRN: http://ssrn.com/abstract=416740.

[248] D. J. Solove. 'I've got nothing to hide' and other misunderstandings of privacy. *San Diego Law Review*, 44, 2007.

[249] L. Spitzner. Honeytokens: The other honeypot. SecurityFocus Infocus technical article (July 17, 2003). http://www.securityfocus.com/infocus/1713.

[250] Statistics Canada. Canadian Internet Use Survey 2007, June 2008. http://www.statcan.gc.ca/daily-quotidien/080612/t080612b-eng.htm.

[251] R. C. Stefan Berger, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: Virtualizing the trusted platform module. In *USENIX Security Symposium*, Vancouver, Canada, 2006.

[252] StopBadware.org. StopBadware.org identifies companies hosting large numbers of websites that can infect internet users with badware. Press release (May 3, 2007). http://www.stopbadware.org/home/pr_050307.

[253] K. Strater and H. Richter. Examining privacy and disclosure in a social networking community (poster). In *Symposium on Usable Privacy and Security (SOUPS'07)*, Pittsburgh, PA, USA, July 2007.

[254] X. Suo and Y. Zhu. Graphical passwords: A survey. In *Annual Computer Security Applications Conference (ACSAC'05)*, Tucson, AZ, USA, Dec. 2005.

[255] Symantec Security Response. Banking in silence. News article (Jan. 14, 2008). http://www.securityfocus.com/blogs/485.

[256] Symantec.com. Symantec global Internet security threat report: Trends for July - December 07. Published in Apr. 2008 (Volume XIII).

[257] The Arizona Republic. Man's ID theft nightmare takes 2 years to iron out. News article (Feb. 13, 2008). http://www.azcentral.com/community/mesa/articles/0213mr-idtheft0214.html.

[258] The New York Times. Antiviral scareware just one more intruder. News article (Oct. 29, 2008). http://www.nytimes.com/2008/10/30/technology/internet/30virus.html.

[259] The Sydney Morning Herald. NZ bank adds security online. News article (Nov. 8, 2004). http://www.smh.com.au/.

[260] TheRegister.com. Phishing attack targets one-time passwords. News article (Oct. 12, 2005). http://www.theregister.co.uk/2005/10/12/outlaw_phishing/.

[261] TheSun.co.uk. Your life for sale. News article (June 23, 2005). http://www.thesun.co.uk/sol/homepage/news/article108989.ece.

[262] Toronto Star. Social networking sites hacker targets. News article (Aug. 3, 2007). http://www.thestar.com/sciencetech/Technology/article/243096.

[263] W. Treese. Once collected, data isn't private. *netWorker*, 9(3), 2005.

[264] TrendMicro.com. Mobile security. http://trendmicro.com/mobilesecurity.

[265] Trusted Computing Group. TCG physical presence interface specification, Apr. 2007. Version 1.00, final revision 1.00, for TPM family 1.2; level 2.

[266] H. Tuch, G. Klein, and G. Heiser. OS verification — now! In *USENIX Hot Topics in Operating Systems (HotOS'05)*, Santa Fe, NM, USA, June 2005.

[267] M. Tulloch. Resolving Windows XP SP2 – related application compatibility problems. Microsoft article on using XP.

[268] US Monitor. Mail monitoring and list protection service. `http://www.usmonitor.com`.

[269] E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. In *Workshop on Usable Security (USEC'07)*, Lowlands, Scarborough, Trinidad/Tobago, Feb. 2007.

[270] P. van Oorschot. Message authentication by integrity with public corroboration. In *New Security Paradigms Workshop (NSPW'05)*, Lake Arrowhead, CA, USA, Sept. 2005.

[271] P. van Oorschot and S. Stubblebine. Countering identity theft through digital uniqueness, location cross-checking, and funneling. In *Financial Cryptography and Data Security (FC'05)*, Roseau, Commonwealth of Dominica, 2005.

[272] M. Vea. 2006 Operating System vulnerability summary. Online article published at OmniNerd.com (Mar. 26, 2007).

[273] Verizon Business Risk Team. 2008 data breach investigations report. Analysis of 500 security breach and data compromise engagements between 2004 – 2007. `http://www.verizonbusiness.com/resources/security/databreachreport.pdf`.

[274] WashingtonPost.com. Hackers zero in on online stock accounts. News article (Oct. 24, 2006).

[275] D. Weinshall. Cognitive authentication schemes safe against spyware (short paper). In *IEEE Symposium on Security and Privacy*, Oakland, CA, USA, May 2006.

[276] D. Weirich and M. A. Sasse. Pretty good persuasion: A first step towards effective password security in the real world. In *New Security Paradigms Workshop (NSPW'01)*, Cloudcroft, NM, USA, Sept. 2001.

[277] C. Wharton, J. Rieman, C. Lewis, and P. Polson. The cognitive walkthrough method: A practitioner's guide. In "Usability inspection methods," John Wiley & Sons, Inc., 1994.

[278] A. Whitten and J. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *USENIX Security Symposium*, Washington, DC, USA, Aug. 1999.

[279] I. Wiener. Sample SecurID token emulator with token secret import, 2000. BugTraq post. `http://archives.neohapsis.com/archives/bugtraq/2000-12/0428.html`.

[280] WindowsSecrets.com. Microsoft, McAfee, Symantec charge cards repeatedly. News article (May 17, 2007).

[281] Wired.com. Citibank replaces some ATM cards after online PIN heist – update. Blog article (June 20, 2008). `http://blog.wired.com/27bstroke6/2008/06/citibank-issues.html`.

[282] Wired.com. Fraudsters target Facebook with phishing scam. News article (Jan. 3, 2008). `http://www.wired.com/politics/security/news/2008/01/facebook_phish`.

[283] Wired.com. LifeLock founder resigns amid controversy. Wired blog article (June 11, 2007). `http://blog.wired.com/27bstroke6/2007/06/lifelock_founde_1.html`.

[284] Wired.com. Private Facebook pages are not so private. News article (June 28, 2007). `http://www.wired.com/software/webservices/news/2007/06/facebookprivacysearch`.

[285] D. Wright, M. Friedewald, W. Schreurs, M. Verlinden, S. Gutwirth, Y. Punie, I. Maghiros, E. Vildjiounaite, and P. Alahuhta. The illusion of security. *Communications of the ACM*, 51(3), Mar. 2008.

[286] M. Wu, S. Garfinkel, and R. Miller. Secure web authentication with mobile phones. In *DIMACS Workshop on Usable Privacy and Security Systems*, July 2004.

[287] M. Wu, R. Miller, and S. L. Garfinkel. Do security toolbars actually prevent phishing attacks. In *Conference on Human Factors in Computing Systems (CHI'06)*, Montreal, QC, Canada, Apr. 2006.

[288] T. Wu. A real-world analysis of Kerberos password security. In *Network and Distributed System Security Symposium (NDSS'99)*, San Diego, CA, USA, Feb. 1999.

[289] G. G. Xie, C. E. Irvine, and T. E. Levin. Quantifying effect of network latency and clock drift on time-driven key sequencing. In *IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'02)*, Vienna, Austria, July 2002.

[290] M. Xie, Z. Wu, and H. Wang. HoneyIM: Fast detection and suppression of instant messaging malware in enterprise-like networks. In *Annual Computer Security Applications Conference (ACSAC'07)*, Miami Beach, FL, USA, Dec. 2007.

[291] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security & Privacy*, 2(5), Sept.-Oct. 2004.

[292] J. J. Yan. A note on proactive password checking. In *New Security Paradigms Workshop (NSPW'01)*, Cloudcroft, NM, USA, Sept. 2001.

[293] Z. E. Ye, S. Smith, and D. Anthony. Trusted paths for browsers. *ACM Transactions on Information and System Security (TISSEC)*, 8(2), 2005.

[294] ZapTheDingbat. Has MasterCard gone on a phishing trip, leaving the back door wide open?, June 2004. `http://www.zapthedingbat.com/security/scriptinjection/`.

[295] ZDNet.com. Security tools face increased attack. News article based on Yankee Group report (June 20, 2005).

[296] ZDNet.com.au. Eighty percent of new malware defeats antivirus. News article (July 19, 2006).

[297] Y. Zhang, S. Egelman, L. F. Cranor, and J. Hong. Phinding phish: An evaluation of anti-phishing toolbars. In *Network and Distributed System Security Symposium (NDSS'07)*, San Diego, CA, USA, Feb. 2007.

[298] M. E. Zurko. User-centered security: Stepping up to the grand challenge. In *Annual Computer Security Applications Conference (ACSAC'05)*, Tucson, AZ, USA, Dec. 2005. Invited essay.

[299] M. E. Zurko and R. T. Simon. User-centered security. In *New Security Paradigms Workshop (NSPW'96)*, Lake Arrowhead, CA, USA, Sept. 1996.

[300] M. Zviran and W. J. Haga. Cognitive passwords: The key to easy access control. *Computers & Security*, 9(8), Dec. 1990.

Appendix A

Online Banking User Survey

In this chapter, we provide the survey questionnaire (as noted in Section 2.4), results of our user survey, and a discussion on the survey data.

A.1 Survey Questionnaire

Below we list the questions used for the survey reported in Section 2.4.

1. Do you use online banking?
   □ Yes □ No Comments ______________________________
   (If No, you don't need to answer the following questions.)
2. Which bank do you use for online banking?
   □ Prefer not to say □ RBC □ CIBC □ TD Canada Trust □ Scotiabank
   □ BMO □ PC Financial □ Other ______________________________
3. What browser do you use for online banking?
   □ Internet Explorer 6 (IE6) □ IE7 □ Firefox □ Mozilla □ Netscape □ Opera
   □ Safari □ Konqueror □ Other ______________________________
4. What operating system (OS) do you use for online banking?
   □ Windows □ Mac □ Linux □ Linux LiveCD □ Don't know
   □ Other ______________
5. Do you keep your operating system (OS) up-to-date with security patches?
   □ Yes, by □ No □ Don't know Comments ______________
   □ automatic update
   □ manual update
   □ don't know
6. Do you keep your web browser up-to-date with security patches?
   □ Yes, by □ No □ Don't know Comments ______________
   □ automatic update
   □ manual update
   □ don't know
7. Do you have the following anti-malware tools in some or all computers you use for online banking?

   (a) Anti-virus: □ Yes on all □ Yes on some □ No □ Don't know
   (b) Firewall (software/hardware): □ Yes on all □ Yes on some □ No □ Don't know
   (c) Anti-spyware: □ Yes on all □ Yes on some □ No □ Don't know

8. Do you keep your anti-malware tools up-to-date with updates and security patches?
   □ Yes, by □ No □ Don't know Comments ________
   - □ automatic update
   - □ manual update
   - □ don't know

9. On the same computers that you use for online banking:
   - (a) Do you run file-sharing or P2P software, e.g., bittorrent, eMule, KaZaA?
     □ Yes □ No □ Don't know Comments ________
   - (b) Do you use Windows file sharing (e.g., sharing files on LAN, default is ON) on them?
     □ Yes □ No □ Don't know Comments ________

10. When you are finished with an online banking session which of the following do you do promptly:
    - (a) Sign-out from your bank: □ Yes □ No □ Don't know Comments ________
    - (b) Clear the browser cache: □ Yes □ No □ Don't know Comments ________
    - (c) Close the browser: □ Yes □ No □ Don't know Comments ________

11. How frequently do you change your online banking password?
    □ Monthly □ Yearly □ Don't change □ Don't know □ Other ________

12. How often do you check your bank statements?
    □ Weekly □ Monthly □ Don't check □ Don't know □ Other ________

13. Did you read your banking agreement, privacy and security policies of your bank?
    □ Yes □ No □ Don't know □ Other ________

14. Do you use a unique password (i.e., not related to your other passwords) for online banking?
    □ Yes □ No □ Don't know □ Other ________

15. Do you use unique personal verification questions and answers for online banking?
    □ Yes □ No □ Don't know □ Not applicable □ Other ________

16. All major Canadian banks provide 100% reimbursement guarantee in case of online frauds, if you comply with their policy. If you know them, state up to three major conditions that your bank requires you to fulfill to be eligible for such reimbursements.

    (a)________ (b)________ (c)________

A.2 Survey Data and Discussion

The following tables provide our survey results. We also discuss these findings below (see Section 2.4 for a summary).

	RBC	CIBC	TD	Scotiabank	BMO	PC Financial	Other
No. of users	24	12	28	32	13	17	12

Table A.1: Users per bank

	Browser				Operating System			
	IE6/IE7	Firefox/Mozilla	Safari	Other	Windows	Mac	Linux	Linux LiveCD
Users	33	102	7	5	95	13	34	2
%	23	69	5	3	66	9	24	1

Table A.2: Browser and OS usage

	Anti-virus			Firewall			Anti-spyware		
	Yes	No	Some	Yes	No	Some	Yes	No	Some
Users	61	27	14	77	14	10	45	35	15
%	60	26	14	76	14	10	47	37	16

Table A.3: Anti-malware usage

	OS		Browser		Anti-malware	
	Yes	No	Yes	No	Yes	No
Users	114	12	118	6	85	18
%	90	10	95	5	83	17

Table A.4: Maintaining an up-to-date system

	P2P file sharing		Windows file sharing	
	Yes	No	Yes	No
#	62	42	45	56
%	60	40	45	55

Table A.5: File sharing from the banking PC

	Sign-out		Clear cache		Close browser	
	Yes	No	Yes	No	Yes	No
#	99	7	32	66	55	48
%	93	7	33	67	53	47

Table A.6: Actions at the end of a banking session

	Unique passwd		Unique PVQs	
	Yes	No	Yes	No
#	71	32	56	37
%	69	31	60	40

Table A.7: Unique passwords and PVQs

	Read agreement			State 3 conditions			
	Yes	No	Other	None	One	Two	Three
#	31	68	6	93	5	6	6
%	29	65	6	85	5	5	5

Table A.8: Agreement and requirement awareness

	Password change		Bank statement check	
	Within a year	Don't change	Within a month	Don't check
No. of users	32	70	100	2
% of users	31	69	98	2

Table A.9: Password change and bank statement check frequency

Discussion on the survey results. About 93% of participants (115 of 123) reported using online banking (but note that several users refused to participate in the survey, potentially biasing this statistics). One participant who does not use online banking, commented that he/she "read the agreement and thought it [online banking] too risky; it is impossible to comply with the conditions." Another wrote "too many requirements to ensure. I don't trust the bank to pay up if something goes wrong." Other comments for not using online banking were "do not trust it," "too insecure" etc. We conducted this survey between Jan. and Apr. 2007.

Table A.1 lists the number of users per bank. Many users reported to have accounts with multiple banks, which implies these users must maintain several unique passwords, PINs, and (optionally) PVQs. Most participants use Firefox/Mozilla on Windows (Table A.2) – indicating a technically-biased survey group. Many participants use multiple web browsers and/or operating systems for online banking. Although Firefox/Mozilla is very popular, Scotiabank and PC Financial do not list it as a recommended browser. All banks support Netscape, but none in our survey reported using it. Two-thirds of IE users use IE6 even after months of the release of IE7, i.e., many users do not use the latest *secure* browser version as recommended for online banking. (Note that IE7 is a "critical update" according to Microsoft.) Linux is used by almost a quarter of the participants; two of them use Linux LiveCD. Banks do not explicitly mention support of Linux (except RBC) or LiveCD, although these are apparently better choices for secure OS.[1] Using LiveCD may seem paranoid, and shows users' lack of trust of commodity operating systems (which may be justified as regular OS installations are commonly infested with several forms of malware). Linux and Mac users may find it difficult to comply with banks' anti-malware requirements as there are only few anti-virus and anti-spyware products for those platforms.

Table A.3 summarizes anti-malware use. Most users (76%) have a firewall on all machines that they use for online banking, while 10% do not use any firewall and 14% use firewall on some machines. Less than half of the users always use anti-spyware on machines used for banking. More than a quarter of the participants do

[1] Although banks recommend Windows and Mac as preferred OS, one analysis [272] reported that before patched, (*out-of-box* version), both Windows XP and Mac OS X offer attackers more remotely exploitable vulnerabilities than Linux variants.

not use anti-virus at all. Most users also keep their OS, browser and anti-malware updated (Table A.4). We also collected statistics on update mechanisms (auto/manual). Many users use both automatic and manual updates (we added them together), and some use automatic notification but manual update. Auto updates are used by 70% (OS), 77% (browser), and 74% (anti-malware) of the users who keep their systems updated. One user does not update the OS or browser but relies on a firewall for protection against network attacks. Another updates "only if forced." Some users do not update their firewall as it requires a firmware upgrade of a home router. One user commented that updating anti-malware is "a pain." Around half of the users use P2P software and/or Windows file sharing (Table A.5) against some banks' recommendations. However, a few users mentioned that they do not run P2P clients while performing online banking. One user reported to use an admin account for online banking, and a regular user account for running P2P.

Only few users do not sign-out from online banking when they are done (Table A.6). One user even reported to reboot the PC after a banking session. However, compliance with clearing the browser cache (one-third) or closing the browser (just over one half) is pretty low. Two users mentioned using the auto clear cache feature of Firefox. Closing the browser after a banking session is being "rather too paranoid" according to one user.

A significant portion of the users do not use unique passwords or PVQs (31% and 40% respectively), and 69% of users do not change their password (Tables A.7 and A.9). Only four users reported changing their banking password every month. Apparently most users, and more specifically PC Financial users, do not follow the frequent password change recommendation. One user's comment about PVQs was "I hate those questions." Another commented that "I hope I will remember them" (cf. [232]).

65% of the participants did not read any banking agreements (Table A.8), although all banks assume their customers have "have read and agreed" to all related banking policies when users sign on to online banking. Several users commented that they only skimmed through these agreements. One mentioned reading the agreements, "but did not understand [those] at all." Another reported these agreements and policies as “too complicated to understand.” One participant stopped using online banking as a result of carefully reading the online banking agreement.

85% of the participants (Table A.8) were unable to state any major conditions for being eligible for the 100% reimbursement guarantee.[2] Only six users (5%) could state three conditions although some of those were not accurate; two of them mentioned to be aware of these conditions as they were present in our previous class talk. Several participants answered as “not a clue,” “no idea,” “impossible conditions to achieve” etc. One mentioned “use their credit card” as a requirement (we did not count such answers as valid). We believe that participants could easily state three conditions directly from the questionnaire if they had read online banking agreements; note that 29% of the participants claimed to have read the banking agreements, and thus we believe participants over-reported this item.

Most users apparently check their bank statements within a month. Several check their statements weekly or even daily, although we did not ask “how diligently.” Two participants mentioned not checking their statements. Note that all banks require users to check statements carefully and to report any errors promptly. If a fraudulent transaction is not reported within a certain period (generally 30 days), banks may refuse any reimbursement.

[2] Some participants may have simply been too lazy to answer; we cannot tell as we held no follow-up interviews.

Appendix B

Security Analysis of MP-Auth

In addition to the preliminary security analysis in Section 3.3, in this chapter we analyze MP-Auth using the AVISPA [14] protocol analysis tool, and the Protocol Composition Logic (PCL) [64, 117, 216] proof technique.

B.1 AVISPA Test Code

We include here results of our AVISPA [14] analysis of an idealized version (see below) of the MP-Auth protocol from Section 3.2.

Protocol Purpose

MP-Auth attempts to achieve authentication and key exchange between a mobile device M and a remote server S. More specifically, the protocol goals are (see Section 3.2, Table 3.1 for notation):

- M and S achieve mutual authentication (using P and E_S)
- M and S establish a secret (symmetric) session key for later use

How We Tested Using AVISPA

We used AVISPA Web interface available at `http://www.avispa-project.org/web-interface/`. We copied the HLPSL code (below) to the Web interface, and ran the relevant tests. Applicable tests to MP-Auth are: On the Fly Model Checker (OFMC), Constraint Logic-based Attack Searcher (CL-AtSe), and SAT-based Model Checker (SATMC). The Tree Automata based on Automatic Approximations for Analysis of Security Protocols (TA4SP) results are omitted from the AVISPA output below as the TA4SP back-end was not supported for our setup.

Idealization of MP-Auth

In MP-Auth, the browser B acts like a relaying party between M and S during the authentication and key exchange phase. Therefore B was removed from our idealized HLPSL model (and thus also, the SSL encryption between B and S). Also, the human user U was merged with M, as U only provides the password P to M. Hence the idealized MP-Auth is a two-party protocol, which is much simpler to analyze for AVISPA back-end protocol analyzers. As we have omitted party B, session ID verification is not required. The transaction integrity confirmation messages use K_{MS} established in the authentication phase. The confirmation messages have not been included in our model; we assume the secrecy of K_{MS} implicitly protects those messages. The idealized version of MP-Auth is given below.

```
M <- S: Rs
M -> S: {Rm}_Es.{f(Rs).M.P}_Kms, where Kms = f(Rs.Rm)
M <- S: {f(Rm)}_Kms
```

Results of the AVISPA Tests

No attacks have been reported by AVISPA on the idealized protocol. Results from the AVISPA back-end protocol analyzers are given below.

OFMC.

```
% OFMC
% Version of 2006/02/13
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfileP2NEkh.if
GOAL
  as_specified
BACKEND
```

```
  OFMC
COMMENTS
STATISTICS
  parseTime: 0.00s
  searchTime: 2.58s
  visitedNodes: 798 nodes
  depth: 10 plies
```

CL-AtSe.

```
SUMMARY
  SAFE
DETAILS
  BOUNDED_NUMBER_OF_SESSIONS
  TYPED_MODEL
PROTOCOL
  /home/avispa/web-interface-computation/./tempdir/workfileP2NEkh.if
GOAL
  As Specified
BACKEND
  CL-AtSe
STATISTICS
  Analysed   : 5548 states
  Reachable  : 3529 states
  Translation: 0.01 seconds
  Computation: 0.14 seconds
```

SATMC.

```
SUMMARY
  SAFE
DETAILS
  STRONGLY_TYPED_MODEL
  BOUNDED_NUMBER_OF_SESSIONS
  BOUNDED_SEARCH_DEPTH
  BOUNDED_MESSAGE_DEPTH
PROTOCOL
  workfileP2NEkh.if
GOAL
  %% see the HLPSL specification..
BACKEND
  SATMC
COMMENTS
STATISTICS
  attackFound               false     boolean
  upperBoundReached         true      boolean
  graphLeveledOff           4         steps
  satSolver                 zchaff    solver
  maxStepsNumber            11        steps
  stepsNumber               5         steps
  atomsNumber               1196      atoms
  clausesNumber             5705      clauses
  encodingTime              1.12      seconds
  solvingTime               0.1       seconds
  if2sateCompilationTime    0.21      seconds
ATTACK TRACE
  %% no attacks have been found..
```

HLPSL Specification

```
role mobile (M, S: agent,
      Es: public_key,
      F, KeyGen: hash_func,
      P: text,
      SND, RCV: channel (dy)) played_by M def=

  local State : nat,
  Rm, Rs: text,
  Kms: message
  init State := 1

  transition
    2.  State  = 1 /\ RCV(Rs') =|>
        State':= 3 /\ Rm' := new()
                   /\ Kms':= KeyGen(Rs'.Rm')
                   /\ SND({Rm'}_Es.{F(Rs').M.P}_Kms')
                   /\ witness(M,S,rm,Rm')
                   /\ secret(Kms', sec_kms1, {M,S})

    3.  State  = 3 /\ RCV({F(Rm)}_Kms) =|>
        State':= 5 /\ request(M,S,rs,Rs)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role server(S: agent,
      Es: public_key,
      F, KeyGen: hash_func,
      Agents: (agent.text) set,
      SND, RCV: channel (dy)) played_by S def=

  local State : nat,
  Rm, Rs, P: text,
  Kms: message,
  M: agent

  init State := 0

  transition
    1.  State  = 0 /\ RCV(start) =|>
        State':= 2 /\ Rs' := new()
```

```
                   /\ SND(Rs')

    2. State  = 2  /\ RCV({Rm'}_Es.{F(Rs).M'.P'}_KeyGen(Rs.Rm'))
                   /\ in(M'.P', Agents) =|>
       State':= 4  /\ Kms' := KeyGen(Rs.Rm')
                   /\ SND({F(Rm')}_Kms')
                   /\ secret(Kms', sec_kms2, {M',S})
                   /\ request(S,M',rm,Rm')
                   /\ witness(S,M',rs,Rs)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role session(M, S: agent,
       Es: public_key,
       F, KeyGen: hash_func,
       P: text,
       Agents: (agent.text) set) def=

  local SS, RS, SM, RM: channel (dy)

  composition
     mobile (M,S,Es,F,KeyGen,P,SM,RM)
  /\ server (S,Es,F,KeyGen,Agents,SS,RS)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

role environment() def=

    local Agents: (agent.text) set
    const m, s: agent,
    es: public_key,
    f, keygen: hash_func,
    rm, rs, sec_kms1, sec_kms2 : protocol_id,
    pm, pi: text

    init Agents := {m.pm, i.pi}
    intruder_knowledge = {m,s,f,keygen,pi,es,rs}

    composition
        session(m,s,es,f,keygen,pm,Agents)
     /\ session(m,s,es,f,keygen,pm,Agents)
     /\ session(i,s,es,f,keygen,pi,Agents)
end role

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

goal

  secrecy_of sec_kms1, sec_kms2
  authentication_on rm
  authentication_on rs

end goal

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

environment()
```

B.2 A PCL Proof Sketch for MP-Auth

In this section we discuss a proof sketch of MP-Auth using the Protocol Composition Logic (PCL) [64, 117, 216]. We assume that readers are familiar with the PCL proof system. See Appendix B.2.5 for a quick reference to frequently-used PCL axioms, rules, and definitions. We first outline the PCL setup for MP-Auth, and then provide the PCL analysis of mutual authentication and key secrecy.

B.2.1 PCL Setup

For the proof here, we use the following simplified version of MP-Auth. As the browser in MP-Auth only forwards messages between the web server and personal device, we remove the browser's role here. For simplification of the proof, we also replace $\{f(R_M)\}_{K_{MS}}$ with $[1]_{K_{MS}}$ (i.e., now the proof of ownership of the session key K_{MS} is provided through a MAC instead of an encryption). Also, to reduce confusion between PCL roles (generally upper case) and variables (generally lower case), we make necessary case transformation here.

$$M \leftarrow S : ID_S.r_s$$

$$M \rightarrow S : \{r_m\}_{E_S}.\{f(r_s).ID_U.P\}_{k_{ms}}, \text{ where } k_{ms} = f(r_s.r_m)$$

$$M \leftarrow S : [1]_{k_{ms}}$$

This simplified protocol is defined by 'roles' {**Init**, **Resp**} in Fig. B.1, written using the protocol programming language as used in PCL. Each role specifies a sequence of actions to be executed by an honest principal in MP-Auth. An honest principal can execute one or more copies of its own role concurrently. Note that, roles are asymmetric in MP-Auth; for example, the server and client authenticate each other using a public key and a password, respectively, and an honest server does not impersonate a client. Here, a *thread* X refers to a principal $\hat{X}$ executing a particular instance of a role. Actions inside a thread include nonce generation, encryption, hash calculation, network communication and pattern matching (e.g., decryption). Each thread contains one or more 'basic sequences.' A basic sequence is a series of actions excluding any blocking actions (e.g., receive) except as the first action. Each role in MP-Auth consists of two basic sequences. PCL proofs use modal formulas of the form $\psi[P]_X\varphi$ which informally means that if X starts from a state where ψ holds, and executes the program P, then the resulting state is guaranteed to hold the security property φ, irrespective of the actions of a Dolev-Yao attacker and other honest principals. Let $idp := \langle \hat{M}.ID, \hat{M}.P \rangle$ (i.e., the userid-password pair of the user operating the mobile device $\hat{M}$), and $ids := \hat{S}.ID$ (the server's ID).

```
$\mathbf{Init} = (\hat{M}, S, idp, ids)$ [
        new $r_s$;
        send $\hat{S}.\hat{M}.ids.r_s$;
        receive $\hat{M}.\hat{S}.t$;
        match $t/\langle encrm, encidp \rangle$;
        $r_m$ := pkdec $encrm, \hat{S}$;
        $k_{ms}$ := hash $r_s.r_m$;
        $decval$ := symdec $encidp, k_{ms}$;
        match $decval/\langle hrs', idp \rangle$;
        match $idp/\langle \hat{M}.ID, \hat{M}.P \rangle$;
        $hrs$ := hash $r_s$;
        match $hrs'/hrs$;
        $mac1$ := hash $1, k_{ms}$;
        send $\hat{S}.\hat{M}.mac1$;
    $]_S$
```

```
$\mathbf{Resp} = (M, idp)$ [
        receive $\hat{S}.\hat{M}.ids.r_s$;
        new $r_m$;
        $encrm$ := pkcnc $r_m, \hat{S}$;
        $hrs$ := hash $r_s$;
        $k_{ms}$ := hash $r_s.r_m$;
        $symterm$ := $hrs.idp$;
        $encidp$ := symenc $symterm, k_{ms}$;
        send $\hat{M}.\hat{S}.encrm.encidp$;
        receive $\hat{S}.\hat{M}.mac1$;
        verifyhash $mac1, 1, k_{ms}$;
    $]_M$
```

Figure B.1: MP-Auth server (**Init**) and client (**Resp**) programs

Let $\mathcal{K} = \{\bar{k}_{\hat{S}}\}$, the private key of server $\hat{S}$. The public and private key pair for $\hat{S}$ is $(k_{\hat{S}}, \bar{k}_{\hat{S}})$. We use the following abbreviations for MP-Auth messages (see Fig. B.1 for terms definitions):

$$msg1 := \hat{S}.\hat{M}.ids.r_s$$

$$msg2 := \hat{M}.\hat{S}.encrm.encidp$$

$$msg3 := \hat{S}.\hat{M}.mac1$$

Invariants. The 'honesty' rule in PCL is "an invariance rule for proving properties about the actions of principals that executes roles of a protocol" [64]. An honest principal in PCL is the one who follows one or more roles of the protocol. The honesty rule is used to reason in a deductible manner about the actions of the other party in the protocol. Formulas derived by the application of this rule are called 'invariants'. We use the following invariants of MP-Auth for our authentication and secrecy proofs.[1]

Γ_{mp1}	$\texttt{Honest}(\hat{S}) \wedge \texttt{Send}(S, msg) \supset \neg\texttt{Contains}(msg, idp)$
Γ_{mp2}	$\texttt{Honest}(\hat{M}) \supset \texttt{PkEnc}(M, r_m, k_{\hat{S}}) \supset (\texttt{Receive}(M, msg1) < \texttt{New}(M, r_m) < \texttt{PkEnc}(M, r_m, k_{\hat{S}}) < \texttt{Send}(M, msg2))$
Γ_{mp3}	$\texttt{Honest}(\hat{M}) \wedge \texttt{Receive}(M, msg1) \wedge \texttt{Send}(M, msg2) \supset \texttt{FirstSend}(M, r_m, msg2)$
Γ_{mp4}	$\texttt{Honest}(\hat{M}) \wedge \texttt{Send}(M, msg) \supset \neg\texttt{Contains}(msg, HASH[k_{ms}](1))$
Γ_{mp5}	$\texttt{Honest}(\hat{S}) \wedge \texttt{New}(S, r_s) \wedge \texttt{Send}(S, msg1) \supset \texttt{FirstSend}(S, r_s, msg1)$

Γ_{mp1} states that the server $\hat{S}$ does not send any message containing *idp*. This essentially prohibits a server to execute the role of a client (mobile device). Otherwise, $\hat{S}$ could impersonate $\hat{M}$ which is false given that $\hat{S}$ is honest. Γ_{mp4} implies that an honest $\hat{M}$ does not send any message containing $HASH[k_{ms}](1)$, although $\hat{M}$ also knows k_{ms}. Only the server $\hat{S}$ sends such a term to prove the knowledge of k_{ms}.

Secrecy of password. As assumed in MP-Auth, the userid-password pair *idp* is unique for each user, and P is a shared secret between M and S. This assumption is formalized as follows.

$$\phi_{secp} ::= \texttt{Honest}(\hat{M}) \wedge \texttt{Honest}(\hat{S}) \wedge \texttt{Has}(\hat{Z}, idp) \supset (\hat{Z} = \hat{M} \vee \hat{Z} = \hat{S})$$

Additionally, we now show that *idp* is not sent in the clear by any role (**Init, Resp**) of MP-Auth. The proof is straightforward: $\hat{S}$ does not send any message with *idp*, and $\hat{M}$ sends out *idp* only encrypted under k_{ms}. Assume that $\mathcal{K}' = \{k_{ms}\}$, and k_{ms}

[1] For the application of the honesty rule, the invariants must be preserved by all the basic sequences in MP-Auth. These proofs are straightforward, and thus we omit them here.

is the secret session key shared between $\hat{M}$ and $\hat{S}$ (see Section B.2.3).

$$\textbf{SAF2} \quad \texttt{SafeMsg}(E_{sym}[k_{ms}](hrs.idp), idp, \mathcal{K}') \tag{B.1}$$

$$B.1, \textbf{P1, S1, NET3} \quad \texttt{SafeNet}(idp, \mathcal{K}')[\textbf{Resp}]_M \; \texttt{SendsSafeMsg}(M, idp, \mathcal{K}') \tag{B.2}$$

$$\Gamma_{mp1}, \textbf{SAF0} \quad \texttt{SafeNet}(idp, \mathcal{K}')[\textbf{Init}]_S \; \texttt{SendsSafeMsg}(S, idp, \mathcal{K}') \tag{B.3}$$

From B.2, B.3, and the application of the **NET** rule and the **POS** axiom, we conclude that $\texttt{SafeNet}(idp, \mathcal{K}')$ is always true, and $\texttt{Honest}(\hat{M}) \wedge \texttt{Honest}(\hat{S}) \wedge \texttt{Has}(\hat{Z}, idp) \supset (\hat{Z} = \hat{M} \vee \hat{Z} = \hat{S})$.

Security properties of MP-Auth. MP-Auth requires the following authentication and secrecy properties to be satisfied by any successful protocol run.

1. *Server-side authentication.* At the end of a protocol run, both parties must agree on each other's identity, protocol completion status, and the secret session key generated from exchanged nonces. The authentication property of MP-Auth is formulated in terms of matching conversations [29]. The basic idea is that on execution of the server role (**Init**), we prove the existence of the intended client role (**Resp**) with a corresponding view of the messages exchanged. Matching conversations for server $\hat{S}$ and corresponding client $\hat{M}$ is formulated as follows:

$$\begin{aligned} \phi_{auth,\hat{S}} \; ::= \; & \texttt{Honest}(\hat{M}) \wedge \texttt{Honest}(\hat{S}) \supset \exists M.\texttt{Has}(M, k_{ms}) \wedge \\ & ((\texttt{Send}(S, msg1) < \texttt{Receive}(M, msg1)) \wedge \\ & (\texttt{Receive}(M, msg1) < \texttt{Send}(M, msg2)) \wedge \\ & (\texttt{Send}(M, msg2) < \texttt{Receive}(S, msg2)) \wedge \\ & (\texttt{Receive}(S, msg2) < \texttt{Send}(S, msg3))) \end{aligned}$$

   Note that the server receives no acknowledgement for the last message sent; i.e., the corresponding receive action is not a part of the authentication guarantee.

2. *Secrecy of k_{ms}.* The secret session key K_{ms} must not be known to any principal other than the server and client. This secrecy property of MP-Auth is

formulated as follows:

$$\phi_{seckms} ::= \texttt{Honest}(\hat{M}) \wedge \texttt{Honest}(\hat{S}) \wedge \texttt{Has}(\hat{Z}, k_{ms}) \supset (\hat{Z} = \hat{M} \vee \hat{Z} = \hat{S})$$

3. *Client-side authentication.* In MP-Auth, the client and server roles are not symmetric; i.e., the server is authenticated by showing the proof of ownership of the decryption key corresponding to the public key as used by the client role. On the other hand, the client prove its identity to the server by showing the knowledge of the shared secret P. Thus the client's view of mutual authentication is different than that of the server. For client $\hat{M}$, communicating with server $\hat{S}$, matching conversations is defined as follows.

$$\begin{aligned} \phi_{auth,\hat{M}} ::= \; & \texttt{Honest}(\hat{M}) \wedge \texttt{Honest}(\hat{S}) \supset \exists S.\texttt{Has}(S, k_{ms}) \wedge \\ & ((\texttt{Send}(S, msg1) < \texttt{Receive}(M, msg1)) \wedge \\ & (\texttt{Receive}(M, msg1) < \texttt{Send}(M, msg2)) \wedge \\ & (\texttt{Send}(M, msg2) < \texttt{Receive}(S, msg2)) \wedge \\ & (\texttt{Receive}(S, msg2) < \texttt{Send}(S, msg3)) \wedge \\ & (\texttt{Send}(S, msg3) < \texttt{Receive}(M, msg3))) \end{aligned}$$

B.2.2 Server-side Authentication

We use the secrecy of password (ϕ_{secp}, Section B.2.1), protocol invariants $\Gamma_{mp1}, \Gamma_{mp2}$ and Γ_{mp3}, to argue that there must be a thread of client $\hat{M}$ which must have performed certain actions corresponding to the client role **Resp** (see Fig. B.1). Properties of nonces and encryption are also used. The proof sketch is summarized by the following steps below. Each step consists of three components: (i) the axioms, invariants and/or previous steps used, (ii) actions performed, and (iii) the resulting predicate. For example, the first step of the proof below uses axioms **AA1, P1, AA4** (see Section B.2.5) to establish that the server performed certain actions in sequence.

$$\textbf{AA1, P1, AA4}\quad [\textbf{Init}]_S\ \texttt{Send}(S, msg1) < \texttt{Receive}(S, msg2) < \texttt{Send}(S, msg3) \tag{B.4}$$

$$\textbf{AA1}\quad [\texttt{receive}\ \hat{M}.\hat{S}.t]_S\ \texttt{Receive}(S, \hat{M}.\hat{S}.t) \tag{B.5}$$

$$\textbf{AR1}\quad \texttt{Receive}(S, \hat{M}.\hat{S}.t)[\texttt{match}\ t/\langle encrm, encidp\rangle]_S \\ \texttt{Receive}(S, \hat{M}.\hat{S}.encrm.encidp) \tag{B.6}$$

$$\textbf{REC}\quad \texttt{Receive}(S, \hat{M}.\hat{S}.encrm.encidp) \supset \texttt{Has}(S, \hat{M}.\hat{S}.encrm.encidp) \tag{B.7}$$

$$\textbf{PROJ}\quad \texttt{Has}(S, \hat{M}.\hat{S}.encrm.encidp) \supset \texttt{Has}(S, encrm) \wedge \texttt{Has}(S, encidp) \tag{B.8}$$

$$\textbf{AR3}\quad \texttt{Has}(S, encrm)\ [r_m := \texttt{pkdec}\ encrm, \hat{S};]_S\ \texttt{Has}(S, E[k_{\hat{S}}](r_m)) \tag{B.9}$$

$$\textbf{DEC}\quad \texttt{Has}(S, E[k_{\hat{S}}](r_m)) \supset \texttt{Has}(S, r_m) \tag{B.10}$$

$$B.5, B.6, B.7, B.8, \\ B.9, B.10, \textbf{S1, P1}\quad [\textbf{Init}]_S\ \texttt{Has}(S, r_m) \tag{B.11}$$

$$\textbf{AA1}\quad [\texttt{new}\ r_s]_S\ \texttt{New}(S, r_s) \tag{B.12}$$

$$\textbf{ORIG}\quad \texttt{New}(S, r_s) \supset \texttt{Has}(S, r_s) \tag{B.13}$$

$$B.12, B.13, \textbf{S1, P1}\quad [\textbf{Init}]_S\ \texttt{Has}(S, r_s) \tag{B.14}$$

$$\textbf{HASH0'}\quad [k_{ms} := \texttt{hash}\ r_s.r_m]_S\ \texttt{Has}(S, k_{ms}) \tag{B.15}$$

$$\textbf{AR3}\quad \texttt{Has}(S, encidp)\ [decval :=\ \texttt{symdec}\ encidp, k_{ms};]_S \\ \texttt{Has}(S, E_{sym}[k_{ms}](decval)) \tag{B.16}$$

$$\textbf{AR1}\quad \texttt{Has}(S, E_{sym}[k_{ms}](decval))\ [\texttt{match}\ decval/\langle hrs', idp\rangle;]_S \\ \texttt{Has}(S, E_{sym}[k_{ms}](hrs'.idp)) \tag{B.17}$$

$$\textbf{ENC4}\quad \texttt{SymDec}(S, E_{sym}[k_{ms}](hrs'.idp), k_{ms}) \supset \exists Y.\texttt{SymEnc}(Y, hrs'.idp, k_{ms}) \tag{B.18}$$

$$\textbf{ENC3}\quad \texttt{SymEnc}(Y, hrs'.idp, k_{ms}) \supset \texttt{Has}(Y, hrs'.idp) \wedge \texttt{Has}(Y, k_{ms}) \tag{B.19}$$

$$\textbf{PROJ}\quad \texttt{Has}(Y, hrs'.idp) \supset \texttt{Has}(Y, hrs') \wedge \texttt{Has}(Y, idp) \tag{B.20}$$

$$B.20, \phi_{secp}\quad \texttt{Has}(Y, idp) \supset \hat{Y} = \hat{M} \vee \hat{Y} = \hat{S} \tag{B.21}$$

$$\begin{array}{rl} \Gamma_{mp1}, B.16, B.17, B.18, & \\ B.19, B.20, B.21, \textbf{S1, P1} & [\textbf{Init}]_S\ \exists Y.\texttt{Has}(Y, idp) \supset \hat{Y} = \hat{M} \end{array} \tag{B.22}$$

$$\Gamma_{mp2}, B.22 \quad [\textbf{Init}]_S\ \exists M.(\texttt{Receive}(M, msg1) < \texttt{Send}(M, msg2)) \tag{B.23}$$

$$\textbf{AN3, FS1, S1, P1} \quad [\textbf{Init}]_S\ \texttt{FirstSend}(S, r_s, msg1) \tag{B.24}$$

$$\begin{array}{rl} B.22, B.24, \textbf{FS2} & [\textbf{Init}]_S\ \texttt{Receive}(M, msg1) \wedge \hat{M} \neq \hat{S} \\ & \supset \texttt{Send}(S, msg1) < \texttt{Receive}(M, msg1) \end{array} \tag{B.25}$$

$$\textbf{AA1, S1, P1} \quad [\textbf{Init}]_S\ \texttt{Receive}(S, msg2) \tag{B.26}$$

$$\begin{array}{rl} \Gamma_{mp3}, B.26, \textbf{FS2} & [\textbf{Init}]_S\ \texttt{Honest}(\hat{M}) \wedge \hat{M} \neq \hat{S}\ \wedge \texttt{Receive}(M, msg1) \\ & \wedge\, \texttt{Send}(M, msg2) \supset \texttt{Send}(M, msg2) < \texttt{Receive}(S, msg2) \end{array} \tag{B.27}$$

$$B.4, B.23, B.25, B.27 \quad [\textbf{Init}]_S\ \texttt{Honest}(\hat{M}) \wedge \hat{M} \neq \hat{S} \supset \phi_{auth,\hat{S}} \tag{B.28}$$

B.2.3 Secrecy of Session Key

We show that honest principals do not perform any actions that compromise the secrecy of session key k_{ms} through induction on the basic protocol sequences (see below for definitions of $\textbf{Resp}_1$, $\textbf{Resp}_2$, $\textbf{Init}_1$, and $\textbf{Init}_2$). Each induction step informally states that if k_{ms} has not already been compromised at the beginning of a basic sequence (i.e., $\texttt{SafeNet}(k_{ms}, \mathcal{K})$ is true), then the actions performed in that basic sequence by a thread X do not compromise k_{ms} (i.e., $\texttt{SendsSafeMsg}(X, k_{ms}, \mathcal{K})$ is true). For the basic sequence $\textbf{Resp}_2$, the proof is straightforward: there is no `send` action. For $\textbf{Init}_2$, the terms sent out by $\hat{S}$ do not contain k_{ms} in the clear (k_{ms} is used as a MAC key).

$$\text{Let } [\textbf{Resp}_2]_{M'} : \begin{array}{l} [\texttt{receive}\ \hat{S}'.\hat{M}'.mac1'; \\ \quad \texttt{verifyhash}\ mac1', 1, k'_{ms};]_{M'} \end{array}$$

$$\textbf{S1, NET1} \quad \texttt{SafeNet}(k_{ms}, \mathcal{K})[\textbf{Resp}_2]_{M'}\ \texttt{SafeMsg}(\hat{S}'.\hat{M}'.mac1', k_{ms}, \mathcal{K}) \tag{B.29}$$

$$B.29, \textbf{NET2} \quad \texttt{SafeNet}(k_{ms}, \mathcal{K})[\textbf{Resp}_2]_{M'}\ \texttt{SendsSafeMsg}(M', k_{ms}, \mathcal{K}) \tag{B.30}$$

```
Let $[\mathbf{Init}_2]_{S'}$ : [receive $\hat{M}'.\hat{S}'.t'$;
                 match $t'/\langle encrm', encidp'\rangle$;
                 $r'_m$ := pkdec $encrm'$, $\hat{S}'$;
                 $k'_{ms}$ := hash $r'_s.r'_m$;
                 $decval'$ := symdec $encidp'$, $k'_{ms}$;
                 match $decval'/\langle hrs'', idp'\rangle$;
                 match $idp'/\langle \hat{M}'.ID, \hat{M}'.P\rangle$;
                 $hrs'$ := hash $r'_s$;
                 match $hrs''/hrs'$;
                 $mac1'$ := hash $1, k'_{ms}$;
                 send $\hat{S}'.\hat{M}'.mac1'$;$]_{S'}$
```

$$\textbf{SAF5} \quad \texttt{SafeMsg}(HASH[k'_{ms}](1), k_{ms}, \mathcal{K}) \tag{B.31}$$

$$B.31, \textbf{S1, NET3} \quad \texttt{SafeNet}(k_{ms}, \mathcal{K})[\mathbf{Init}_2]_{S'}\ \texttt{SendsSafeMsg}(S', k_{ms}, \mathcal{K}) \tag{B.32}$$

The session key k_{ms} is computed from two nonces, r_s and r_m, where r_s is sent in the clear. Thus the secrecy of k_{ms} lies on the secrecy of r_m. For the basic sequences that send out nonces, we need to show that the nonces are not equal to r_m, or that r_m is encrypted under the public key of $\hat{S}$. These arguments are formulated as $\Phi := \Phi^1_{r_m} \wedge \Phi^2_{r_m}$.

$$\Phi^1_{r_m} : \forall M, \hat{Z}.\texttt{New}(M, r_m) \wedge \texttt{PkEnc}(M, r_m, k_{\hat{Z}}) \supset \hat{Z} = \hat{S}$$

$$\Phi^2_{r_m} : \forall M, \texttt{New}(M, r_m) \wedge \texttt{Send}(M, msg) \supset \neg\texttt{ContainsOpen}(msg, r_m)$$

The predicate $\texttt{ContainsOpen}(m, a)$ asserts that a can be obtained from m (directly or a series of unpairings only) without any decryption. $\Phi^1_{r_m}$ and $\Phi^2_{r_m}$ can be established from invariant Γ_{mp2}: from thread M's point of view, it knows that it has freshly generated the nonce r_m, and has only sent r_m out encrypted with only principal $\hat{S}$'s public key.

$$\begin{aligned}\text{Let } [\mathbf{Resp_1}]_{M'}: \ & [\mathtt{receive}\ \hat{S}'.\hat{M}'.ids'.r'_s;\ \mathtt{new}\ r'_m;\\ & encrm' \ := \ \mathtt{pkcnc}\ r'_m,\ \hat{S}';\\ & hrs' \ := \ \mathtt{hash}\ r'_s;\ k'_{ms} \ := \ \mathtt{hash}\ r'_s.r'_m;\\ & symterm' \ := \ hrs'.idp';\\ & encidp' \ := \ \mathtt{symenc}\ symterm',\ k'_{ms};\\ & \mathtt{send}\ \hat{M}'.\hat{S}'.encrm'.encidp';]M'\end{aligned}$$

$$\text{Case}: \ r'_m \neq r_m \tag{B.33}$$

$$\mathbf{S1,\ SAF3} \quad [\mathbf{Resp_1}]_{M'}\ \mathtt{SafeMsg}(E_{pk}[\hat{S}'](r'_m), r_m, \mathcal{K}) \tag{B.34}$$

$$B.34, \mathbf{NET3} \quad \mathtt{SafeNet}(r_m, \mathcal{K})[\mathbf{Resp_1}]_{M'}\ \mathtt{SendsSafeMsg}(M', r_m, \mathcal{K}) \tag{B.35}$$

$$\text{Case}: \ r'_m = r_m \tag{B.36}$$

$$\mathbf{S1,\ P1} \quad [\mathbf{Resp_1}]_{M'}\ \mathtt{PkEnc}(M', r_m, k_{\hat{S}'}) \tag{B.37}$$

$$B.37, \Phi^1_{r_m} \quad [\mathbf{Resp_1}]_{M'}\ \hat{S}' = \hat{S} \tag{B.38}$$

$$B.37, B.38, \mathbf{SAF3} \quad [\mathbf{Resp_1}]_{M'}\ \mathtt{SafeMsg}(E_{pk}[\hat{S}](r_m), r_m, \mathcal{K}) \tag{B.39}$$

$$B.39, \mathbf{NET3} \quad \mathtt{SafeNet}(r_m, \mathcal{K})[\mathbf{Resp_1}]_{M'}\ \mathtt{SendsSafeMsg}(M', r_m, \mathcal{K}) \tag{B.40}$$

$$\begin{aligned}\text{Let } [\mathbf{Init}_1]_{S'}: \ & [\mathtt{new}\ r'_s;\\ & \mathtt{send}\ \hat{S}'.\hat{M}'.ids'.r'_s;]_{S'}\end{aligned}$$

$$\Phi^2_{r_m} \quad [\mathbf{Init}_1]_{S'}\ r'_s \neq r_m \tag{B.41}$$

$$B.41, \mathbf{SAF0} \quad [\mathbf{Init}_1]_{S'}\ \mathtt{SafeMsg}(\hat{S}'.\hat{M}'.r'_s, r_m, \mathcal{K}) \tag{B.42}$$

$$B.42, \mathbf{NET3} \quad \mathtt{SafeNet}(r_m, \mathcal{K})[\mathbf{Init}_1]_{S'}\ \mathtt{SendsSafeMsg}(S', r_m, \mathcal{K}) \tag{B.43}$$

As k_{ms} is computed from r_m and r_s, we can say $\mathtt{SafeNet}(r_m, \mathcal{K}) \supset \mathtt{SafeNet}(k_{ms}, \mathcal{K})$. Thus, from B.30, B.32, B.35, B.40, B.43 and the application of the **NET** rule and the **POS** axiom, we conclude that $\mathtt{SafeNet}(k_{ms}, \mathcal{K})$ is always true, and $\Phi \wedge \mathtt{Honest}(\hat{M}) \wedge \mathtt{Honest}(\hat{S}) \wedge \mathtt{Has}(\hat{Z}, k_{ms}) \supset (\hat{Z} = \hat{M} \vee \hat{Z} = \hat{S})$.

B.2.4 Client-side Authentication

We use protocol invariants Γ_{mp4} and Γ_{mp5}, secrecy of the server's private key, and properties of nonces to argue that there must be a thread of server $\hat{S}$ which must have performed certain actions corresponding to the server role **Init** (see Fig. B.1). The proof sketch is summarized by the following steps below.

$$\textbf{AA1, P1, AA4} \quad [\textbf{Resp}]_M\ \texttt{Receive}(M, msg1) < \texttt{Send}(M, msg2) < \texttt{Receive}(M, msg3) \tag{B.44}$$

$$\textbf{AA1} \quad [\texttt{receive}\ \hat{S}.\hat{M}.mac1]_M\ \texttt{Receive}(M, \hat{S}.\hat{M}.mac1) \tag{B.45}$$

$$B.45, \textbf{S1, P1} \quad [\textbf{Resp}]_M\ \texttt{Receive}(M, \hat{S}.\hat{M}.mac1) \tag{B.46}$$

$$B.46, \textbf{HASH3'} \quad [\textbf{Resp}]_M \wedge \texttt{Honest}(\hat{M}) \supset \exists X.\texttt{Computes}(X, HASH[k_{ms}](1)) \wedge \texttt{Send}(X, msg) \wedge \texttt{Contains}(msg, HASH[k_{ms}](1)) \tag{B.47}$$

$$B.46, B.47 \quad [\textbf{Resp}]_M \wedge \texttt{Honest}(\hat{M}) \supset \exists X.(\texttt{Receive}(M, msg3) < \texttt{Send}(X, msg3)) \tag{B.48}$$

$$\textbf{HASH2} \quad [\texttt{verifyhash}\ mac1, 1, k_{ms}]_M\ mac1 = HASH[k_{ms}](1) \tag{B.49}$$

$$B.49, \textbf{S1, P1} \quad [\textbf{Resp}]_M\ mac1 = HASH[k_{ms}](1) \tag{B.50}$$

$$B.50, \Gamma_{mp4}, \textbf{SEC, DEC} \quad [\textbf{Resp}]_M \wedge \texttt{Honest}(\hat{S}) \supset \exists X.((\texttt{Send}(X, msg3) < \texttt{Receive}(X, msg2)) \wedge \texttt{PkDec}(X, E[k_{\hat{S}}](r_m), \bar{k}_{\hat{S}})) \supset (\hat{X} = \hat{S} \wedge \texttt{Has}(S, r_m)) \tag{B.51}$$

$$\textbf{S1, P1, AN3, FS1} \quad [\textbf{Resp}]_M\ \texttt{FirstSend}(M, r_m, msg2) \tag{B.52}$$

$$B.52, \textbf{FS2} \quad [\textbf{Resp}]_M\ \texttt{Receive}(S, msg2) \wedge \hat{M} \neq \hat{S} \supset \texttt{Send}(M, msg2) < \texttt{Receive}(S, msg2) \tag{B.53}$$

$$\Gamma_{mp5}, \textbf{AA1, FS2} \quad [\textbf{Resp}]_M\ \texttt{Receive}(M, msg1) \wedge \hat{M} \neq \hat{S} \supset \texttt{Send}(S, msg1) < \texttt{Receive}(M, msg1) \tag{B.54}$$

$$B.44, B.48, B.51, B.53, B.54 \quad [\textbf{Resp}]_M \wedge \texttt{Honest}(S) \wedge \hat{M} \neq \hat{S} \supset \phi_{auth,M} \tag{B.55}$$

B.2.5 Frequently-used PCL Axioms, Rules, and Definitions in MP-Auth

The PCL axioms and rules that we use here have been proposed previously [64, 117, 216, 144]. Some of these axioms are natural logical assumptions (also known as first order logical axioms, e.g., creation of a nonce implies possession of that nonce). Others are *idealized* cryptographic axioms which provide formal logic equivalent of standard cryptography. (Note that, in reality, most cryptographic primitives do not achieve idealized cryptographic functionality.) In the axioms here, a denotes an action (e.g., `send`, `receive`, `new`, `pkenc`), and `a` denotes the corresponding predicate in PCL. Axiom **AA4** states that after thread X executes actions a, ..., b in a sequence, the action predicates `a`, ..., `b` are temporarily ordered in the corresponding sequence. Axiom **SEC** states that if a principal $\hat{X}$ is honest, and a thread Y of another principal $\hat{Y}$ can decrypt a term encrypted with the public key of $\hat{X}$ then principals $\hat{X}$ and $\hat{Y}$ must be the same ($\wedge$ is logical conjunction and $\supset$ can be read as *implies*). We introduce a new axiom **HASH0'** which refers to the fact that if principal X computes the hash of a value then X also possesses the computed hash. Table B.1 lists the commonly used axioms, rules, and definitions in our analysis.

AA1	$\phi[a]_X\ \texttt{a}$
AA4	$\phi[a;...;b]_X\ \texttt{a} < ... < \texttt{b}$
AN3	$\phi[\texttt{new}\ x]_X\ \texttt{Fresh}(X,x)$
REC	$\texttt{Receive}(X,x) \supset \texttt{Has}(X,x)$
ENC	$\texttt{Has}(X,x) \wedge \texttt{Has}(X,K) \supset \texttt{Has}(X,E[K](x))$
PROJ	$\texttt{Has}(X,x.y) \supset \texttt{Has}(X,x) \wedge \texttt{Has}(X,y)$
DEC	$\texttt{Has}(X,E[K](x)) \wedge \texttt{Has}(X,K) \supset \texttt{Has}(X,x)$
AR1	$\texttt{a}(x)[\texttt{match}\ q(x)/q(t)]_X\ \texttt{a}(t)$
AR3	$\texttt{a}(x)[y := \texttt{dec}\ x, K]_X\ \texttt{a}(E[K](y))$
SEC	$\texttt{Honext}(\hat{X}) \wedge \texttt{Decrypt}(Y, E[k_{\hat{X}}](x)) \supset (\hat{Y} = \hat{X})$
G4	$\frac{\phi}{\theta[P]_X\phi}$
S1	$\frac{\phi_1[P]_X\phi_2 \quad \phi_2[P']_X\phi_3}{\phi_1[PP']_X\phi_3}$
P1	$\texttt{Persist}(X,t)[a]_X\ \texttt{Persist}(X,t)$ for $\texttt{Persist} \in \{\texttt{Has, Send, Receive}\}$
FS1	$\texttt{Fresh}(X,t)[\texttt{send}\ t']_X\ \texttt{FirstSend}(X,t,t')$, where $t \subseteq t'$
FS2	$\texttt{FirstSend}(X,t,t') \wedge \texttt{a}(Y,t'') \supset \texttt{Send}(X,t) < \texttt{a}(Y,t'')$, where $X \neq Y$ and $t \subseteq t''$
ENC3	$\texttt{Enc}(X,m,k) \supset \texttt{Has}(X,k) \wedge \texttt{Has}(X,m)$, where $\texttt{Enc} \in \{\texttt{SymEnc}, \texttt{PkEnc}\}$
PENC4	$\texttt{PkDec}(X, E[k](m), \bar{k}) \supset \exists Y.\texttt{PkEnc}(Y,m,k)$
ENC4	$\texttt{SymDec}(X, E_{sym}[k](m), k) \supset \exists Y.\texttt{SymEnc}(Y,m,k)$
HASH3'	$\texttt{Receive}(X, HASH[k](x)) \supset$ $\exists Y.\texttt{Computes}(Y, HASH[k](x)) \wedge \texttt{Send}(Y,m) \wedge \texttt{Contains}(m, HASH[k](x))$
HASH2	$\phi[\texttt{verifyhash}\ m', m, k]_X\ m' = HASH[k](m)$
HASH0'	$\texttt{Computes}(X, HASH(m)) \supset \texttt{Has}(X, HASH(m))$
SAF0	$\neg\texttt{SafeMsg}(s,s,\mathcal{K}) \wedge \texttt{SafeMsg}(x,s,\mathcal{K})$, where x is an atomic term, and $x \neq s$
SAF2	$\texttt{SafeMsg}(E_{sym}[k](m), s, \mathcal{K}) \equiv \texttt{SafeMsg}(m,s,\mathcal{K}) \vee k \in \mathcal{K}$
SAF3	$\texttt{SafeMsg}(E_{pk}[k](m), s, \mathcal{K}) \equiv \texttt{SafeMsg}(m,s,\mathcal{K}) \vee \bar{k} \in \mathcal{K}$
SAF5	$\texttt{SafeMsg}(HASH[k](m), s, \mathcal{K})$
	$\texttt{SendsSafeMsg}(X,s,\mathcal{K}) \equiv \forall m.(\texttt{Send}(X,m) \supset \texttt{SafeMsg}(m,s,\mathcal{K}))$
	$\texttt{SafeNet}(s,\mathcal{K}) \equiv \forall X.\texttt{SendsSafeMsg}(X,s,\mathcal{K})$
NET1	$\texttt{SafeNet}(s,\mathcal{K})[\texttt{receive}\ m]_X\ \texttt{SafeMsg}(m,s,\mathcal{K})$
NET2	$\texttt{SendsSafeMsg}(X,s,\mathcal{K})[\texttt{a}]_X\ \texttt{SendsSafeMsg}(X,s,\mathcal{K})$, where $\texttt{a}$ is not a $\texttt{send}$
NET3	$\texttt{SendsSafeMsg}(X,s,\mathcal{K})[\texttt{send}\ m]_X\ \texttt{SafeMsg}(m,s,\mathcal{K}) \supset \texttt{SendsSafeMsg}(X,s,\mathcal{K})$
POS	$\texttt{SafeNet}(s,\mathcal{K}) \wedge \texttt{Has}(X,m) \wedge \neg\texttt{SafeMsg}(m,s,\mathcal{K}) \supset \exists k \in \mathcal{K}.\texttt{Has}(X,k) \vee \texttt{New}(X,s)$

Table B.1: Frequently-used PCL axioms, rules, and definitions

Appendix C

Review of Earlier PIN Cracking Attacks

For convenience to the reader and for reference within, here we summarize several representative attacks from Berkman and Ostrovsky [32]. For reasons of brevity, we omit how some specific assumptions required by these attacks are met, as well as any efficiency analysis of these attacks (e.g., how many API calls are required for a given attack to succeed).

C.1 Translate PIN Block Attacks

We review the translate-only API attack which requires an attacker to generate/collect Encrypted PIN Blocks (EPBs) of all possible PINs, and access to the translate API function. This attack reveals plaintext PINs, and can be applied at a switch or verification facility. The steps in the attack are as follows.

1. Let A_x be any attacker chosen PAN.
2. Attackers collect/generate $10,000$ EPBs which pack all possible PINs in any ISO format (i.e., the format and PAN of those EPBs are immaterial). Suppose i is any 4-digit PIN, and E'_i packs i in any ISO format.
3. Translate all $10,000$ EPBs to ISO-0 EPBs using A_x as the PAN. Assume E_i is the resulting EPB from the translation API.

   $$E_i = \mathsf{Translate}_{\mathsf{ISO-0}}(E'_i, A_x), \text{where } i \in \{0000 \ldots 9999\}.$$

   Now E_i packs PIN i in the ISO-0 format (with respect to A_x). Make a table with the resulting EPBs and PINs, i.e., (E_i, i).
4. For any customer EPB, E_c, calculate

   $$E_t = \mathsf{Translate}_{\mathsf{ISO-0}}(\mathsf{Translate}_{\mathsf{ISO-1}}(E_c), A_x).$$

Here, an attacker first converts the customer EPB to ISO-1 (which unlinks a PIN with the corresponding customer PAN), and then uses this result with the attacker's chosen PAN to generate an EPB in ISO-0 format.

5. Locate E_t in the table generated at step 3. The corresponding PIN is the PIN packed inside E_c.

C.2 Attacks Exploiting the IBM Calculate-Offset API

The steps in the IBM Calculate-Offset attack at a verification facility and intermediate switch are now outlined.

Calculate-offset attacks at a verification facility. Here the attacker is someone at a verification facility, e.g., an application developer. The steps in the attack are as follows.

1. Generate an EPB E_a that packs a known Final PIN PF_a.
2. For any customer account, A_c, calculate:

$$\mathit{offset} = \mathsf{CalculateOffset}(E_a, A_c).$$

If the customer's Natural PIN is PN_c, then *offset* $= PF_a - PN_c$. Here '$-$' is digit by digit modulo 10 subtraction; *offset* and PF_a are known to the attacker. Thus the attacker learns the customer's Natural PIN. If the attacker can read the plaintext offset value of the customer, then the customer's Final PIN is revealed.

Calculate-offset attack at a switch. The steps of a calculate-offset attack at a switch are as follows.

1. Generate an EPB E_a that packs a known Final PIN PF_a.
2. Select any (random) PAN A_x.
3. Assume that attackers do not have access to the real issuer key at a switch. However, they can calculate a dummy offset using a dummy issuer key (i.e., whatever issuer key is available in the switch's HSM):

$$\mathit{offset}_{d1} = \mathsf{CalculateOffset}(E_a, A_x)$$

i.e., $\mathit{offset}_{d1} = PF_a - PN_{xd}$. Here PN_{xd} is the dummy Natural PIN with respect to the account A_x. So now PN_{xd} can be calculated as both PF_a and offset_{d1} are known.

4. For any customer EPB E_c which packs the customer's Final PIN PF_c, calculate:

$$\mathit{offset}_{d2} = \mathsf{CalculateOffset}(E_c, A_x)$$

i.e., $\mathit{offset}_{d2} = PF_c - PN_{xd}$. The value of PN_{xd} is known from the previous step, thus revealing the customer's Final PIN.

C.3 Attacks Exploiting the VISA PIN Verification Value (PVV)

The steps in the VISA PVV attack at a verification facility and intermediate switch are outlined below.

PVV attacks at a verification facility. Attackers need an EPB with a known PIN, and may need write access to the issuer's PVV database. Again, like offset values, PVVs are considered security insensitive. The attack is as follows.

1. Generate an EPB E_a which packs a known Final PIN PF_a.
2. For any customer PAN A_c,

$$pvv = \mathsf{CalculatePVV}(E_a, A_c).$$

3. Use the calculated PVV with known PIN to create new bank cards (this may also require updating the PVV database at the verification facility).

PVV attacks at a switch. Using 10,000 EPBs which pack all possible PINs, attackers can reveal candidate PINs (less than two, on average) for any customer as follows. Note that the attack HSM here does not have access to the real issuer PVV key; the attack succeeds if *any* PVV key is available.

1. Choose any PAN A_x.
2. Generate EPBs for all possible PINs; assume E_i packs PIN i, where $i \in \{0000 \ldots 9999\}$.
3. For all EPBs generated in step 2, calculate PVVs with respect to A_x:

   $$pvv_i = \mathsf{CalculatePVV}(E_i, A_x).$$

   Now sort the values of pvv_i and build a table of entries (pvv_i, i). More than one (on average less than two) PINs may be indexed by a given PVV.
4. For any customer EPB E_c, compute

   $$pvv = \mathsf{CalculatePVV}(E_c, A_x).$$

   Use the resulting PVV as an index to the table built in step 3. The corresponding PIN is the customer's Final PIN PF_c; in case of multiple PIN values indexed by pvv, PF_c is one of those values; building the table using a different A_x may resolve collisions.