Technical Report

UCAM-CL-TR-731
ISSN 1476-2986

Number 731

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

A new approach to Internet banking

Matthew Johnson

September 2008

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted July 2008 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Trinity Hall.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

A new approach to Internet banking

Matthew J. Johnson

Summary

This thesis investigates the protection landscape surrounding online banking. First, electronic banking is analysed for vulnerabilities and a survey of current attacks is carried out. This is represented graphically as an attack tree describing the different ways in which online transactions can be attacked.

The discussion then moves on to various defences which have been developed, categorizing them and analyzing how successful they are at protecting against the attacks given in the first chapter. This covers everything from TLS encryption through phishing site detection to two-factor authentication.

Having declared all current schemes for protecting online banking lacking in some way, the key aspects of the problem are identified. This is followed by a proposal for a more robust defence system which uses a small security device to create a trusted path to the customer, rather than depend upon trusting the customer's computer. The protocol for this system is described along with all the other restrictions required for actual use. This is followed by a description of a demonstration implementation of the system.

Extensions to the system are then proposed, designed to afford extra protection for the consumer and also to support other types of device. There is then a discussion of ways of managing keys in a heterogeneous system, rather than one managed by a single entity.

The conclusion discusses the weaknesses of the proposed scheme and evaluates how successful it is likely to be in practice and what barriers there may be to adoption in the banking system.

Contents

List of figures

Chapter 1

Introduction

TODAY'S WORLD IS ONE with increasing online access to services. One part of this which is growing rapidly is online banking. Combined with online retailers there is a lot of money changing hands, directed only by communication over the Internet.

This is very convenient and the ready access to the Internet in all first-world countries, coupled with the cost savings from closing bank branches, is driving the deployment and adoption of these services. Purely online transactions, however, lead to increased risk. None of the normal safeguards of real-world transactions are present. Conversely, risk to the criminals is a lot lower (the attacker can be in a completely separate jurisdiction from all the other parties in the transaction) and the retailer sees nothing but a faceless, nameless connection providing card details.

The economy of scale which the Internet and its millions of connected computers provide also works for the criminals. In the past the attacker would be lucky to target a few tens of people for a lot of effort, meaning that attacks must aim to score big and fairly frequently. Now it is a simple task to target millions of people and a small percentage falling foul of the scam still represents a large return on investment.

As few who use the Internet can have failed to notice this has lead to the birth of the phishing scam and its huge growth. Phishing remains one of the highest profile online attacks against financial institutions. Problems such as pump and dump stock scams have also risen in popularity, but these are not so much attacks on the systems themselves and are harder to combat technically. They are certainly less well publicized.

This rapid growth in the industry has lead, as it always does, to many systems implemented with the focus on deploying the features as soon as possible and little or no thought about security. As such this is a time of flux, with many people trying to develop more robust replacements to replace these early, easy to

attack systems.

Most of the development of online financial services has been reactive, doing the minimum amount of work to try and frustrate the attacks which are observed. It has also been quite piecemeal and uncoordinated. Almost all of the defences have a simple attacker model which only considers those attacks which their prospective target has experienced in the wild. Some of these systems manage to achieve their (fairly limited) goals, but many of them are only partially effective at best.

In reaction to the defensive schemes developed by the targets of attacks, many criminals have started to become more sophisticated. This is still lost in the noise of the remarkably successful but simple attacks, which explains why very few people are working on more robust systems. Nevertheless, these new attacks prove that the criminals can adapt to break the defences which are currently being rolled out.

This thesis is a discussion of the attack and defence landscape surrounding online banking and how these high profile targets and their users can best be protected.

The first two chapters are a discussion of the current state of the art in (known) attacks and defences. This thesis shows that while the state of the art in attacks is very much more sophisticated than simple phishing attacks, they are still sufficiently low profile that few people are considering them. On the flip side, defence mechanisms have almost entirely been built as a reaction to attacks which have garnered interest from the media or target institutions. This has lead to a distinct gap between what can be stopped and what the criminals have available to them.

The novel work which is presented in the remainder of the thesis comprises the introduction and description of a more robust defence of Internet banking. This is followed by the application of the system to Internet shopping and to providing better protection for consumers in the event of disputes with their bank. Much of this work was presented at the The 12th Nordic Workshop on Secure IT-systems held at the University of Reykjavik in October 2007 [1].

In addition to the work presented here, work in similar areas has been published in the Thirteenth, Fourteenth and Sixteenth International Workshops on Security Protocols. These papers discuss the security of multiple roles in personal computing devices [2], dealing with unidentified principals in embedded computing situations [3] and real-world uses of multi-party computations [4]. Since they are not central to the thesis of this work they are not discussed in any more detail.

Chapter 2

An analysis of e-banking vulnerabilities

THIS CHAPTER IS ARRANGED into two main sections. The first details general attack strategies for committing Internet banking fraud. The second section contains specific attack vectors which have been or could be used to carry out these strategies. With each of the strategies there is a link to the vectors which can be used to implement them.

2.1 Vulnerability analysis

The are few common ways of formalizing the analysis of a system for vulnerabilities. The Common Criteria [5] comprise a system which formalizes the process by which products and systems can be validated against a number of "protection profiles". The Common Criteria focus mainly on validating products, but the protection profiles provide some useful notation which will be used below. These profiles and certification of products are almost entirely in textual form.

Schneier introduced the "attack tree" method of describing and analysing the attacks on a system [6]. Attack trees are a form of root-cause analysis [7] which provide a graphical way of describing the security of systems. The basic structure represents attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as paths to leaf nodes. Goals will be something like 'steal money' and this is broken down into the steps required to achieve that goal, getting into more detail further down the tree. Multiple branches at each level may be alternatives or all required to achieve a certain step. A similar graphical notation for attacks was suggested more recently by Jakobsson [8] but with a slightly different aim. Schneier presents a top-down approach to graphing attacks which allows a more systemic analysis. The cost or

difficulty of attacks can also be propagated up the tree to easily see which attacks are the most cost-effective to defend against.

Each of the strategies and vectors discussed in this chapter will be assembled on to an attack graph which will then be used to contrast the various defensive measures currently in use in Chapter 3.

2.2 Attack strategies

There are four main types of attack on e-banking. All of the attacks seen at the moment fall into the first two categories: getting authentication credentials from the victim or modifying the victim's legitimate transactions. The other two attacks are less useful, denying them access to their banking or merely observing the transactions of the victim, but they still merit discussion.

2.2.1 Credential harvesting

A very well-known online fraud in the UK is phishing, which is an attack designed to convince the victim to give away their online banking credentials to a third party. This and other similar scams or attacks which reveal credentials to the attacker fall into the class of credential harvesting.

Vectors: *Trojans (V.Trojan), Keyloggers (V.Keylogger), Social engineering (V.SocEng), "Shoulder surfing" (V.Surf).*

S.Phish Phishing web sites/emails

Phishing web sites deserve a separate section. They are the most commonly seen form of credential harvesting attack in the wild, usually combined with an email which tricks the user into accessing the web site. They are purely a social engineering attack which relies on user misunderstanding of security features and problems with how security indicators are presented to users. A comprehensive text covering phishing attacks and countermeasures is the book 'Phishing and Countermeasures' edited by Jakobsson and Myers [9]. Other good summaries are provided by Ollman [10] and Jakobsson [8]. The latter models phishing attacks in more detail and describes more powerful versions of the attack.

In order to present a web site which the user can be convinced is genuine there must be little to indicate that it is not. Obviously the web site must look like the real one but there are a number of other indicators which must also be made to look authentic. Some of the common tricks used in this are given below.

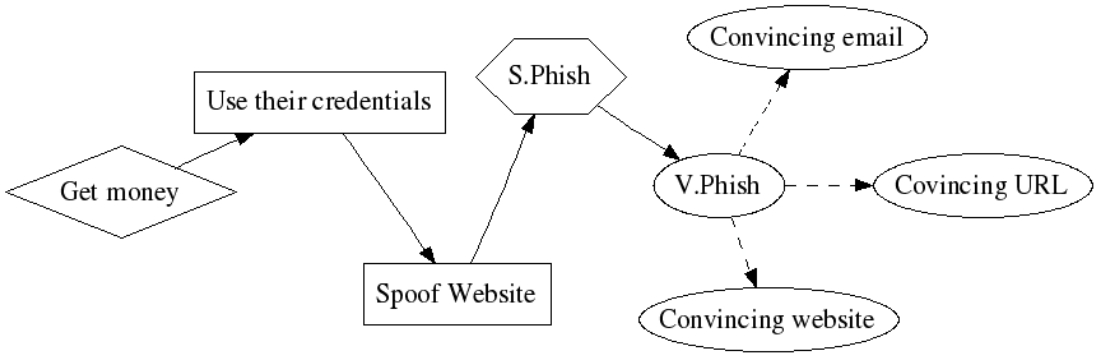

Figure 2.1: Attack graph for S.Phish

Attack graph The attack graph for the phishing attack can be seen in Figure 2.1. The dashed lines represent several things which must all be achieved for a successful attack.

Vectors: *Social engineering (V.SocEng)*

Extended character sets In order to have a successful phishing web site the URL must look plausible. The first technique for doing this is using web browser support for extended character sets in domain names. Homograph attacks were first described in 2002 [11]. The current standard for encoding Unicode characters in a legal URL is the Punycode standard [12] which uses two hyphens as an escape. Such URLs are displayed as the Unicode characters in the address bar and in links. Because Unicode contains several distinct characters which appear very similar (for example 'g' (Unicode 0x0581) and 'h' (Unicode 0x10b9), from the Armenian and Georgian alphabets respectively) it is possible to create a URL which is distinct, but appears the same to the human eye. This was used to attack PayPal [13].

Typo-squatting Typo-squatting is a technique where attackers register similar domains to the target website, either through simple spelling mistakes or common typing errors. These are used to host pornographic website, ads, malware and sometimes phishing attacks. McAfee recently released a report on typo-squatting [14].

Sub-domains A common phishing technique is to rely on the fact that the user cannot tell the difference between similar URLs such as `www.barclays.com` and `www.barclays.secure-banking.com`. The average member of the public will see the word barclays and assume they are owned by the same company, whereas in fact there is no guarantee that this is the case at all. This technique

is helped by the fact that many companies are using URLs outside their normal domain for legitimate sites [15]. This make it almost impossible for a user to tell them apart.

Usernames in URLs HTTP, the web protocol, has built in authentication. Normally this results in a prompt for username and password, however, it also includes support for providing these details in the URL so as to skip the prompt. The URL `http://username:password@www.host.com/` accesses the web site `www.host.com` with a given username and password. The web server can be configured to ignore these and as a result the attacker can craft a username part which looks like the host name but is, in fact, ignored. A sufficiently long username part will result in the real host name not being visible in the address bar.

Image/3-D spam The spam producers are also investing in techniques to foil classifiers which inspect their email. For a long time there has been spam where the real text is included in an image, rather than the email body. There are now reports [16] of spams in which the text is distorted by applying 3-D transformations to make text-extraction from the image more difficult.

S.Vish Vishing

Vishing is both a recent and a very old scam. It is the age old fraud where the attacker phones the victim and uses social engineering to trick the victim into revealing secret information such as credit card information. What is new is the use of voice-over-IP and how this changes the expected trust in the phone system.

Vectors: *Social engineering (V.SocEng)*

Cloned voice-banking systems Many banks have systems for voice-banking. Many vishing attacks clone these systems so that they sound the same as the official systems. Emails similar to those used in phishing attacks solicit customers to call a number purporting to be their bank. Telephone numbers have none of the normal clues to identify their owners so it is very hard for users to distinguish those owned by their bank. This was used to attack Santa Barbara Bank and Trust in 2006 [17].

Voice-over-IP Traditionally the phone service has been a trustworthy source. With caller ID a number could be traced easily to a customer and while phreaking and other attacks were possible, they were quite difficult and specialized. With the advent of voice-over-IP and gateways from IP telephony to the public

switched telephone network associating a number with a real person has become a whole lot harder. Caller-ID is easily spoofed by an attacker and there can be a much more convoluted trail between a VoIP connection and a real person.

Automated answering systems The automated answering and menu systems used by most large companies, including banks, can also be used by an attacker. Combined with VoIP and war-dialling techniques an attacker can automatically try hundreds of numbers and use an automated system which, like banks, solicits details like credit card numbers in the name of ease of use or security. Only once they have a candidate victim who has responded to the automated system do they need to involve a human. Since the hardware to do this is a modern computer, rather than an expensive voice switch, this attack is both scalable and affordable.

S.Inject Traffic injection

Less common, at least in the UK, are attacks which modify transactions being made by the user in order to redirect funds or change the amounts concerned. Traffic injection is an attack which has been around for a while. Traditionally this is done by hacking a router through which the traffic passes, manipulating the Internet routing systems or forging packets. Since this attack has been around for a while many defences have been implemented against it; a review of several of these is given in Chapter 3.

As a result, it is actually one of the least common attacks in practice. Recently, however, there have been a number of new attack vectors seen which bypass some or all of the traditional defences and are a lot easier to do in practice. Of particular note is the attack described in Section V.Trojan. Trojans can inject and rewrite traffic after it has passed through all the traditional defence mechanisms against traffic alteration and is therefore very effective.

Vectors: *Evil Tor nodes (V.Tor), proxy servers (V.Proxy) and access points (V.Wap); hacking Internet routers (V.Router), and ADSL routers (V.ADSL); Trojans (V.Trojan).*

S.Pharm Pharming

Pharming [18] is a specific phishing technique where the attacker alters the DNS responses to a client computer causing a legitimate URL to resolve to the IP of a machine under the control of the attacker.

Vectors: *Evil proxy servers (V.Proxy), evil public access points (V.Wap), hacking Internet routers or DNS servers (V.Router), DNS Poisoning (V.DNS), hacking ADSL*

routers (V.ADSL), Trojans (V.Trojan).

S.DoS Denial of service

Any attack which stops the user from carrying out legitimate transactions can be considered denial of service. Often attacks in Section S.Inject also result in denial of service.

S.Snoop Transaction snooping

Merely being able to read the transaction log of the victim doesn't seem like much of an attack at first glance. However, it is still an invasion of privacy and exact details of some transactions have been used as an authentication mechanism for other services [19]. Any system for protecting online banking needs to at least consider this form of attack.

2.3 Attack vectors

The sections below each correspond to a specific attack vector used in Internet banking fraud.

V.Surf Shoulder surfing

"Shoulder surfing" is the term for surreptitiously observing someone entering credentials in person, usually by looking over their shoulder. This attack vector is normally associated with observing the personal identification number (PIN) for a bank card prior to stealing the physical card either by force or by pickpocketing it.

This is usually either an opportunistic attack or a very targeted, specific one. It certainly does not scale very well in either case and is quite high risk. Someone closely connected to the thief must be physically close to the person while they are entering the PIN.

A more sophisticated variant uses closed-circuit television to observe the PIN. This is less likely to be caught, but more difficult to set up. Depending on the amount of insider help required for installation, it might also be more damaging for the insider if caught. This can be combined with a card skimmer attached to whatever the card is inserted in to, in order to produce nearly automated card duplication and PIN observation.

The latter modifications allow for some scaling of the attack and in this form it has been seen on automated teller machines (ATMs) [20, 21] and in a number of petrol stations [22]. The latter being an insider attack and the former third-party

tampering. It still, however, does not benefit from the economies of scale of the Internet and has quite a high level of risk.

Shoulder Surfing not only applies to PIN entry, but also to credentials on online banking. Typically, these are entered in the privacy of one's own home, but sometimes people log into online banking from public locations, often Internet cafés. Here both electronic (typically insider) observation and opportunistic physical observation are possible as discussed above. These attacks are easier than PIN-based ones as there is normally no need to steal or clone a physical token. However, in this case there are other insider attacks which are both easier and more powerful described in Sections V.Trojan and V.Keylogger.

V.Keylogger Hardware keyloggers

If an attacker has physical access to a machine then they can use a hardware keylogger. These devices are produced commercially [23] and are very cheap and easy to disguise, typically being inserted between the keyboard and the back of the computer, which people rarely look at.

One obvious place for these to be useful is on public computers, such as in Internet cafés. They may be more expensive and difficult to use than just installing a Trojan, but in the cases where the software may be monitored for Trojans, or the attacker is an outsider and doesn't have administrative privileges on the machines they may still be an option. Since they capture all keyboard input they require some processing of the data to find any credentials.

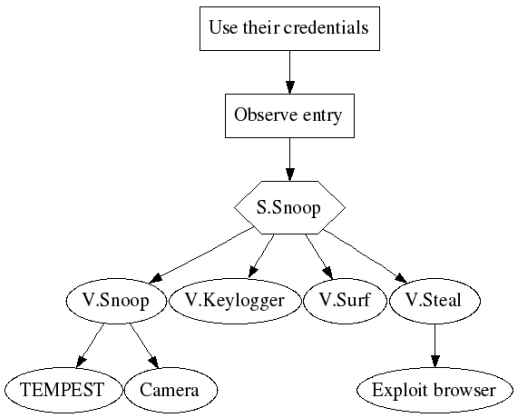

Figure 2.2: Attack graph for V.Surf/V.Keylogger

Pure keyloggers are defeated by some of the simple schemes using the mouse to input credentials rather than the keyboard. See Section 3.4.5 for how this fails if the logger can also log other things.

Attack graph The attack graphs for both shoulder surfing and keyloggers are very similar. Figure 2.2 combines the two graphs.

V.SocEng Social engineering

Social engineering is a broad category which covers every case where the attacker tries to convince the target to do something they should not. In theory all social engineering attempts should be noticed by the target (and hence fail), but in prac-

tice they can be very subtle and hard to differentiate from something legitimate, even for a trained professional.

V.Tor Evil Tor nodes

Tor [24] is an anonymization system advocated by the Electronic Freedom Foundation. It is targeted at a wide variety of people from corporate whistle blowers to dissidents in totalitarian regimes. Various military organizations are using it to conceal whether units have been deployed. It originally used a classic onion routing scheme with the originating node encrypting the real packet in all the layers for the full delivery path under the key of each node in turn. Recent versions use nested SSL connections to achieve the same effect.

What is often not realized is that while the data is encrypted in transit, unless application-layer security is used, then it is in the clear at the exit Tor node. If that node is corrupt it is free to observe or modify all of the traffic. While the route within Tor is selected by the entry point and so a corrupt Tor node cannot solicit extra traffic through itself, it can opportunistically intercept any traffic which does go via it. Since Tor is designed to be robust against evil nodes (in terms of anonymity) there is little control on who can add a server to the network, making this attack quite easy. Tor acknowledge this and say that all traffic via Tor should use application-level authentication and encryption, but many people do not do this. While Tor also blocks nodes which are discovered to be doing attacks, there is a large window in which a Tor node can be evil. Once discovered, the attacker can move the evil node to another server.

This was seen in a recent attack on Tor [25] in which an evil exit node altered HTML replies to submit forms to a server local to the originating computer. In this case it allowed the attacker to access the local Tor server running on the originating machine (normally restricted to connections from localhost and therefore often not password protected) to redirect all traffic via the evil node, increasing the effectiveness of the attack.

This technique is similar to that used in the router hacks in Section V.ADSL and can be used for just that. In effect it provides an easy method to do traffic snooping and injection attacks, without gaining access to a computer on the normal route of a machine. It is not very targeted to specific victims, but will scale up as the amount of traffic over Tor grows.

V.Proxy Public web proxy/anonymizers

As with Tor, there is also a number of public web proxies (often under the guise of anonymizers) which are often used to circumvent local access rules or monitoring. These are normally found by their users through Internet searches, or lists

of proxies with no real control over who is added. There is no system of trust for the people who run these.

If an attacker can solicit the traffic they wish to alter through their proxy, they can redirect any requests and observe any non-TLS traffic. Just using TLS is not sufficient, however, if the user does not check the certificates (users rarely check certificates, or care about them). The attacker could redirect the connection to their own site and then freely observe/modify the traffic.

HTTP redirection (rather than just rewriting) can be used in order to avoid the problem of mismatched TLS certificates. Most victims are unlikely to notice they have been redirected elsewhere and if they do notice most will still assume it is legitimate.

V.Wap Evil public access points

Wireless networking has proliferated at a great rate over the last few years. There are few public places these days not covered by some sort of wireless network. These are run by coffee shops, airports, on trains and even by some city councils[1].

When most people visit these locations they set their laptops to connect to the first access point they find and if this works without a problem ask no further questions. Laptops tend to automatically connect to access point (AP) with the strongest signal strength. If an attacker sets up a rogue access point then a proportion of the people nearby will connect to that access point instead. This can be improved by adding boosters and antennae which are technically illegal, and so won't be used by the official APs, but probably won't be noticed as such and boost the power of the attacker's access point.

Given that some wireless cards can be run in infrastructure mode (such as the Intersil Prism chipset cards[2]), such an access point could appear to be just another commuter using a laptop and so be very unobtrusive. For a customer it could be very difficult to distinguish from the real access point. The conventional 802.11 security mechanisms do not help here either. Symmetric keys mean that if an attacker can access the wireless they can also run an access point for it.

Once the victim has associated with the evil access point, all traffic can be sniffed or modified by the attacker.

V.Router Hacking en-route servers

The traditional traffic observation and modification attack is through compromise of a router between the target machine and the destination of the traffic

[1] `http://www.wififreespot.com/`
[2] `http://hostap.epitest.fi/`

which is to be intercepted. A lot of traditional defences have focussed on this sort of attack, but it is typically quite a difficult and high-risk attack.

Because of routing variations it is more fruitful the closer to the target the compromised router is positioned as more traffic to the target is likely to pass through it. This limits the choice of useful targets for hacking. There are a number of routers in large bottle-necks such as LINX, but these often have a high enough throughput that it is difficult to do packet inspection on them.

Since a lot of security work has focussed on this sort of attack, routers tend to be well-maintained and kept fairly secure through the operating system's defences. Breaches also tend to be well investigated, increasing the risk to the attacker if a compromise is discovered.

V.DNS DNS poisoning

DNS poisoning is a technique by which an attacker inserts bogus entries into the cache of a recursing name server. These entries are then served up to the users of that name server.

Next Generation Security Software's paper on pharming [18] has a good summary of several techniques for DNS poisoning. There are also papers by Wessels [26] and Steinhoff et al [27] along with security advisories covering problems in specific DNS servers [28, 29].

DNS poisoning allows an attacker to redirect connection to the target domain to a machine they control, making injection and modification attacks possible. This is the basis of the pharming attack.

V.ADSL Local router hacks

The closest router to the target is usually the ADSL modem/router which connects subscribers to the Internet via an Internet service provider (ISP). These are typically locked down to deny any connections from the Internet and only accept connections from the local network. This leads manufacturers and users to be lax about changing the default passwords. Because the administration interface is nearly always web-based, this opens up some attacks.

As Stamm et al. [30] have shown, if Javascript can be executed in the user's web browser, either through traffic injection as in the Tor attack above, through cross-site scripting attacks or hacks on legitimate sites or by soliciting traffic to your own web site, these scripts can use the web browser's image loading mechanism and form submission to send traffic to the router.

Full network scans can be achieved through this mechanism, as well as fingerprinting the router to find the model and lookup the default password. This then allows the attacker to reconfigure the router to redirect either just DNS queries or

full traffic via an attacker controlled machine, leading to injection and pharming attacks.

Very recently a comprehensive analysis of many types of attack on home ADSL router has been published by Gnucitizen [31]. This includes some of the general vulnerabilities covered in more detail below, as well as more traditional security issues affecting specific models of router.

Javascript Grossman presented a paper in 2006 [32] on using Javascript to hack internal web servers from an external site. This was followed up in 2007 [33] with an extended attack. The attacker can provide code to be run on the client in the form of Javascript. Normally this would be limited to accessing the original host, but it is possible to work round this by using Javascript to generate `<img>` tags. Such tags are permitted to access other sites. The Javascript can inspect the error state of the browser after generating each image tag. This error state reveals whether the resource given in the image tag exists. If these resources are on other hosts local to the target browser this is, in effect, a scan of the local network for targets.

DNS rebinding There are several systems for running server-provided code on client machines. These generally restrict which hosts can be contacted by the code on the client machine to stop the sort of attacks described in this section. However, both Dean [34] and Jackson [35] have found ways to circumvent these restrictions.

The flaw which a DNS rebinding attack exploits is that these restrictions are designed to cope with multi-homed addresses and hence restrict based on host name. Specifically, they restrict based on the host name which the code was loaded from.

In the DNS rebinding attack, the attacker manipulates the DNS of their domain so that when the code is loaded it points to the correct host name, then later points to an address within the network of the target. This bypasses the security restrictions of most implementations for running server-provided code.

An extreme form of the rebinding attack was demonstrated in 2007 by Kaminsky [36] in which arbitrary network traffic can be sent from a virtual network interface on the attacker's computer to the victim's web browser causing the attacker's computer to appear as if it were on the victim's internal network.

HTML form attacks The HTML form attack introduced by Topf [37] and extended by Eye on Security researchers [38] involves an attacker presenting the user with a form which rather than submitting to the attacker's server, submits to another server and TCP port within the trusted network. This could be done

using any of the other vectors in this chapter to solicit or rewrite traffic. The attacker crafts form data which when sent to the application running on a certain port will cause it to do something bad.

Since the browser sends an HTTP request containing the form data, rather than just the form data, not all services can be attacked this way. Internal web-based services are vulnerable, which many appliances are deploying. The other category of service which is vulnerable are those which ignore input it does not understand. Protocols like POP and SMTP will return an error message when they don't understand the input but then accept further input from the same transaction. This was also the basis of the Tor attack cited in Section V.Tor.

Universal plug and play Universal plug and play is a standard by which devices on a home network can seamlessly integrate and cooperate with each other. One of the common uses of UPnP is automatic configuration of Internet gateway devices to allow connections to and from devices inside the network. In January 2008 Gnucitizen published both an attack on a specific home router [39] and also a more generic attack on home gateways using flash and UPnP [40].

V.Trojan Trojans/worms/viruses

Malicious software installed on the target's computer is a very large category. Trojans, worms and viruses all fit in this category, the difference being the infection vector. This thesis is not concerned with how a computer would become infected with such malicious software, save that there are many instances of this [41, 42, 43] and it is not a very difficult task. Targeting specific individuals may be more tricky, but the economy of scale due to the Internet and the laws of chance suggest that finding a susceptible target will not be hard, if the attacker is not concerned with targeting specific individuals.

The Trojan horse (or just Trojan) is the most powerful weapon in the attacker's collection. It can perform the tasks of many of the other attacks discussed here, such as keyboard logging, traffic interception and rerouting as well as many others.

It is also the most difficult to defend against. Most of the protection mechanisms in use today assume that the user's computer is trusted, in part because most of them are based around software running on the user's computer. Trojans break this assumption and hence everything which relies on it. Defences and how they fail in the presence of Trojans are discussed in Chapter 3.

Attack graph The utility of the Trojan can be seen in the relevant attack graph in Figure 2.3. As can be seen, unlike the very narrow path of the phishing attack

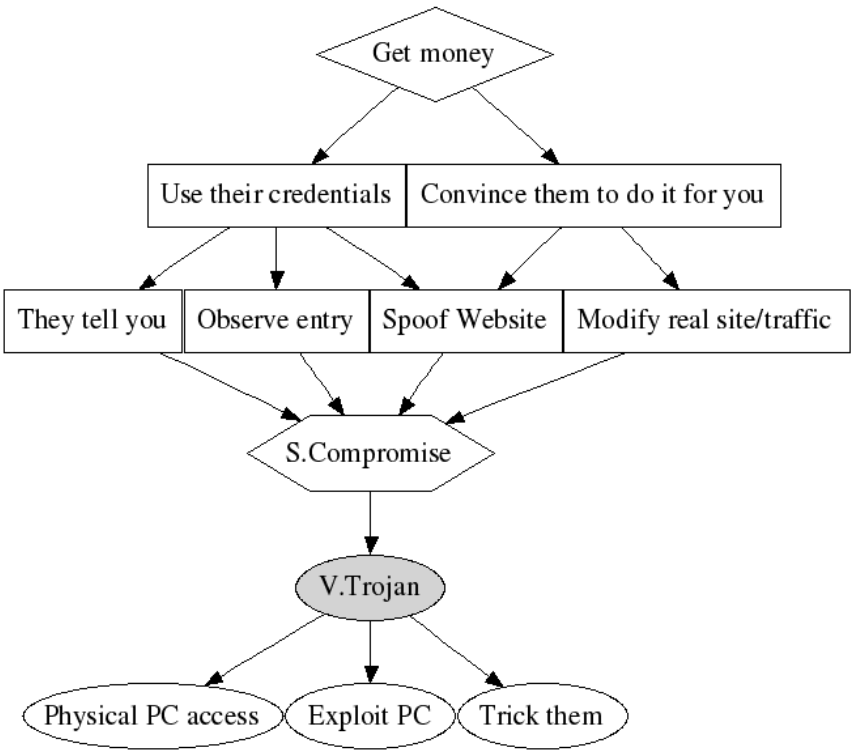

Figure 2.3: Attack graph for V.Trojan

(see Figure 2.1) there are many routes to installing a Trojan and many uses of it. This makes it both very powerful and very versatile

2.4 Attack trees

To protect against an attack one of the required steps for the attack to work needs to be prevented (although the defence in depth principle suggests that it is better to prevent more than one of them). If there are alternative routes for the attack all need to be blocked.

2.4.1 Attack weights

Each node can be annotated with difficulties, costs or other metrics to allow the defender to achieve protection goals such as "protect against any attack which costs less than £100,000". It might not be possible to protect against all possible means of achieving an attack goal, but it might be sufficient to protect against ones which can be afforded or achieved by your predicted attacker.

Cost

One of the obvious metrics is cost. When considering the security of a system it is often useful to consider the value of the target in comparison to the amount spent to protect it and the amount it costs to attack it. It would be foolish to spend £100,000 to defend a target worth only £1000. In addition, an attacker is unlikely to spend £100,000 to attack it; unless they derived some non-monetary benefit from it.

Annotating the attack tree with cost can be used to decide which attacks can be feasibly defended against and which ones are feasible for the attacker.

Access requirements

There are a number of attacks which require the attacker to start with some sort of access privileges before beginning the attack. This is very common and a lot of banking security procedures deal with the problems that insiders pose.

Attacks which require an insider do, however, rule out a number of classes of attacker. They are also more risky for the attacker as once discovered, the target knows a lot about the insider.

Technical complexity

The other obvious metric is the technical complexity of the attack. While this is a lot harder to quantify, it is still very important, particularly when not all attacks can be prevented and priorities must be set over which defences are added. There is no benefit in protecting against obscure complex attacks when other, simpler ones exist.

Amortizable cost

Some attacks require an initial investment which can then be spread over a large number of attacks at little or no extra cost. The economies of scale provided by the Internet make this very feasible. It might cost thousands of pounds to rent a large botnet to send phishing emails, but if millions of emails can be sent then the cost per attack is small. This amortization can be applied not only to money, but also other metrics which might weight the tree.

On the other hand, attacks which require physical intervention in the process are a lot harder to scale and the cost does not amortize over very many instances of the attack.

Amortized cost is not quite the same as a low per-attack cost. Some attackers may not be able to ever afford the initial outlay (particularly if it is not cost, but

rather insider privileges or technological ability) so won't be able to perform the attack at all.

2.4.2 E-banking attack tree

The attack vectors above have been compiled in Figure 2.4.

At the top of the graph are the attack goals in diamonds. These are the goals that the criminal is trying to achieve and are broken down slightly with the square nodes. At the bottom of the graph are the circular nodes, each representing one of the technologies available to the attacker. These correspond to the vectors above.

The hexagons represent the strategies used by attackers which are described at the start of this chapter. An attack strategy is a path through the graph from a goal to a leaf node.

For convenience the attack route from figure 2.1 used in traditional phishing attacks has been highlighted. As can been seen, this is a very small part of the whole attack tree. In contrast, the node which has been shaded in gray is the node representing the use of a Trojan. There are a large number of routes which use this node and thus it is very fruitful for an attacker to exploit.

Annotating the graph is something an organization should do before deciding what attacks they wish to concentrate on. For the purpose of this thesis it is enough to note that in a large number of cases, installing a Trojan is cheap, easy and scales very well. This makes it a very useful and desirable form of attack to the criminals. This attack diagram will be returned to in Chapter 3 to show how well current defences stop attacks.

It should be noted that this is the attack tree considering the Mafia as the attacker. It assumes all the normal parties in the transaction (the merchant, the customer and the bank) are trusted and that attacks come from a third party. As is discussed in Chapter 5, this is not always the case. Each of the parties involved in a transaction may wish to consider the attack tree in which any of the other parties may be complicit in the attacks. This will change the attack tree and add extra paths through it.

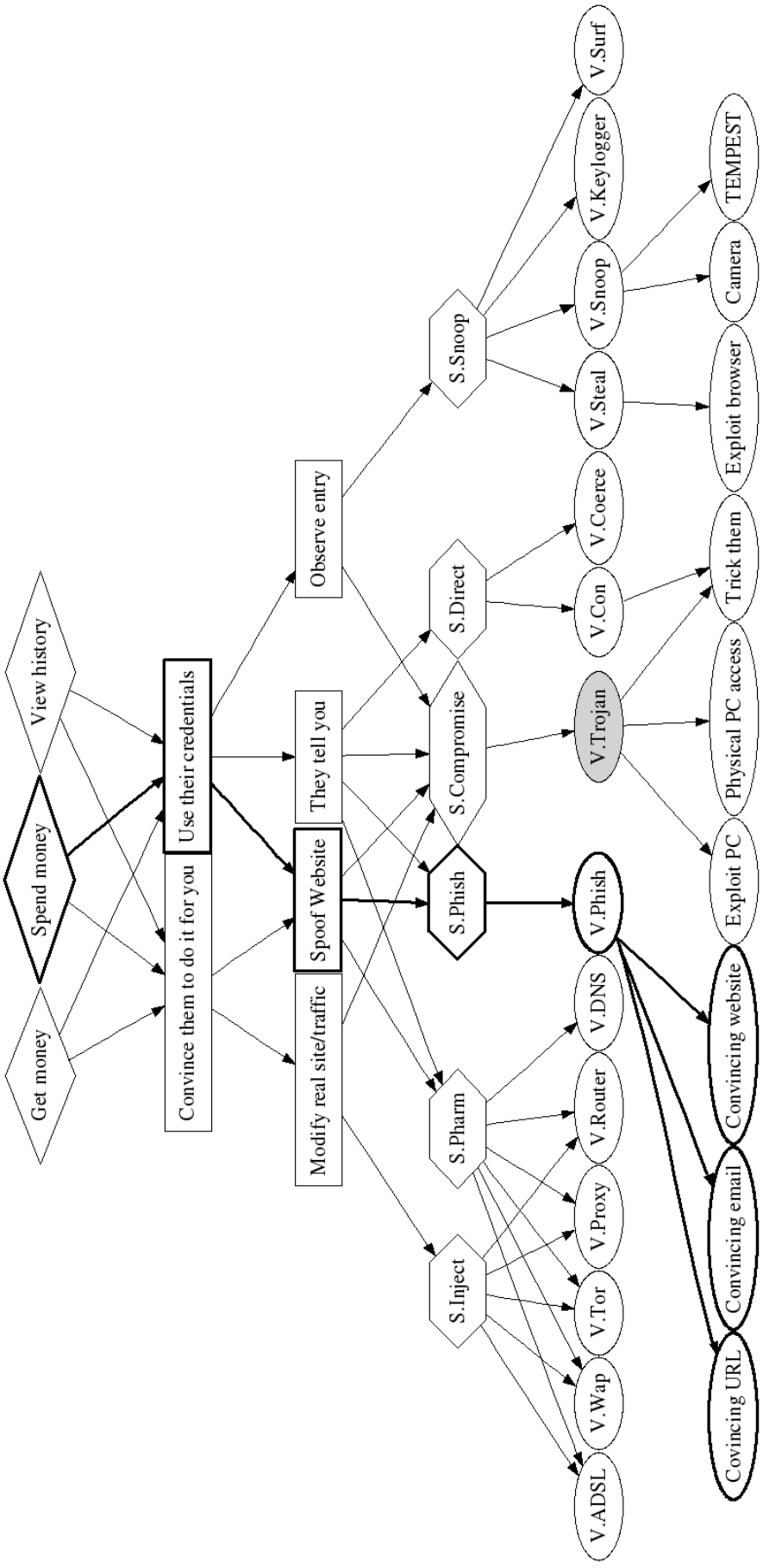

Figure 2.4: Online banking attack tree

Chapter 3

A taxonomy of anti-phishing measures

BECAUSE PHISHING is such a wide-spread threat many people have been deploying or proposing solutions. This chapter reviews the available measures used to combat phishing attacks and evaluates their effectiveness.

3.1 General defences

There is some technology which is in general use outside of just the Internet banking field which can be used to secure online transactions.

3.1.1 TLS

TLS [44] is a transport-layer security protocol which uses a key-agreement protocol such as Diffie-Hellman [45] to establish a confidential channel with integrity guarantees between two parties. Authentication of the server is provided by an X509 certificate chain rooted in one of several trusted third parties whose certificates are provided to clients out-of-band. There is a mechanism for similar certificates to provide client authentication, but very few sites use it and very few users have an appropriate certificate.

TLS is an established and widely used mechanism in the Internet, particularly in the World Wide Web. Most financial or other sensitive data is protected using TLS while in transit across the Internet.

TLS provides several security features. The confidentiality- and integrity-assured channel prevents a number of eavesdropping and data manipulation attacks as well as unsophisticated middle-person attacks. The Diffie-Hellman exchange ensures that this is the case as long as the two ends of the channel have been authenticated. In theory the certificate chain authenticates the remote

server to the client, guaranteeing that the client is communicating with the desired party. Since client certificates are not common, most services employ their own plain-text password protocol within the confidential channel to authenticate the client.

The theoretical certificate chain guarantees unfortunately do not work well in practice. Because the authentication is designed to be mostly transparent and involves concepts unknown to the average user, it is easy to avoid needing a valid certificate. Firstly, the use of TLS is optional on the web. This is exacerbated by the user interface design of most web browsers which display the presence of TLS with a positive indicator, the result of which is that people rarely notice its absence [46]. The same study shows that because of the extensibility of browser platforms, it is often possible to provide fake TLS indicators which will pass even close scrutiny.

In addition, due to the costs associated with acquiring TLS certificates many sites have certificates which are not rooted in the trusted certificates shipped with the browser. When presented with such a certificate browsers will typically prompt the user to accept the certificate. Such prompts do not provide information which will allow the majority of users to make an appropriate security decision since they do not understand the terms used. In such cases the user, who just wants access to the content, has the habit of accepting any question they are asked if it will allow them to do what they want to do.

The other problem with the user understanding TLS certificates is that even when they are valid and rooted in a trusted third party, it is not always clear what they are authenticating. If a user connects to `www.barclays.com` and the certificate is OK, all they can be sure of is that one of their trusted third parties has issued a certificate for `www.barclays.com` to the person controlling the corresponding private key. In the case of `www.barclays.com` there is unlikely to be a mistake in issuing the certificate to someone other than Barclays (although it has happened [47]), but several banks are using uncommon domain names for their official products [48]. The user cannot distinguish between `personalhsbc.co.uk` (a phishing web site) and `www.securesuite.co.uk` (a banking outsourcing firm). Criminals can easily get real certificates for those domains signed by a trusted third party.

The end to end confidentiality and integrity of TLS works very well. Unfortunately, there are a number of ways around the authentication of the other party which open it up to many attacks which are easier than those prevented by the secure channel provided by TLS.

3.1.2 Spam filtering

There are many forms of unsolicited bulk email (spam) and it is a large problem in itself. This has lead to a number of 'classifiers' [49, 50] which try and analyse features of email to assign a score corresponding to the likelihood of the email being spam. This score is used to discard mail whose score is above a given threshold. Since for a lot of phishing attacks initial contact is performed via unsolicited email claiming to be from a bank these are a potential target for being discarded by a spam classifier.

Spam classifiers work by trying to identify features in which legitimate email and spam differ. This is well known to be a hard task for a number of reasons. Firstly, software cannot know in the general case what email a user does not want to receive. Much email which the user wants to read they do not know in advance that they are going to receive.

Spam classifiers usually rely on the authors of spam making mistakes when generating the email, spam sources, or matching specific phrases for well-known products which are sold using spam. Attackers have responded to spam classifiers by altering how they send spam. Compromised machines are used to send spam for short periods of time before moving to a new source machine. This does not give black list-based classification enough time to update to block the email source. Analysis of the email content is worked around by using the same technology used to classify text, to generate text which will pass through filters. Other tricks, such as embedding the spam contents in images which the classifier cannot parse, make blocking spam very difficult. Fundamentally, if the attacker can test messages against the classifier first it is possible for them to work around any defences which are in place.

Banks are also compounding the problem. Several banks have sent official emails out to users which solicit the user to log into web pages [48, 15]—in at least one case, when questioned, the support staff couldn't say whether it was an official site. Distinguishing between phishing email and these can be very hard for an experienced human, let alone a computer or an inexperienced user.

Finally, email is not the only vector for phishing attacks. Vishing, pharming and many other advanced techniques bypass the need for unsolicited email as the first contact point.

3.1.3 Password wizards

All major web browsers today give the option to remember passwords for web sites and to auto-complete the log-in forms so that the user does not have to remember the password. Traditionally, this has been regarded as a security vul-

nerability, since the password can be recovered from where it has been stored in the web browser.

However, there may be cases where it improves security. The web browser is much better than the user at distinguishing whether a site is the same site that it has visited previously. While the user might be tricked by URLs which are similar, but not the same, the web browser will not be. If the user only ever logs in using the saved password then it will be obvious when they visit a phishing web site, since the browser will not offer to complete the form.

This depends on a number of things. Firstly, if banks and other sites use multiple URLs, particularly those outside their normal domain, for logging into the system the user will have to retype their password for legitimate sites and therefore will not be able to spot phishing sites. This is a rule which banks have broken on several occasions. Secondly, the users need to be disciplined enough to trust that the reason the auto-completion is not offered is because this is a fraud, not because the web browser has a fault. Lastly, it assumes that users will always use the same computer. When using Internet cafés or other public terminals they will have to retype their passwords. These are the places most likely to have some sort of keylogger or other malware.

This scheme will only work against primitive phishing attacks. More complicated pharming attacks, or those involving Trojans, will continue to work and, in some cases, will be made easier.

3.1.4 Take down

One of the main defences today is the completely reactive one of take down. There are a number of groups monitoring phishing emails and web sites. These are then reported to the ISPs in question and removed.

By definition there will be some lag between a new phishing web site being released and the site being removed in which victims can fall foul of the scam. This is not the only problem, however. A number of ISPs have been lax at responding to take-down requests [51], which increases the window for successful scams.

Some phishing gangs, notably Rockphish [52] have developed phishing attacks using a large number of web sites hosted on compromised machines. Thus, each email can be sent out with a different link to the host which makes the observers' work much harder. Removing the address in one of the emails doesn't stop many people falling for the scam and removing all the addresses mean collecting a large number more emails. This attack and is described in detail by Jakobsson [53].

Of course, there are other methods of online banking fraud. In the case of Trojans compromising computers directly, take down is useless.

3.2 Web browsers

Since the main part of a phishing attack happens through web browsers, the leading browser manufacturers have all come up with defences in the latest versions of their products. All of the software solutions are ineffective against compromise of the user's computer, but some of them combat other threats.

3.2.1 Microsoft phishing filter

The most recent version of Microsoft Internet Explorer[1], Version 7, comes with built-in anti-phishing measures in the form of the Microsoft Phishing Filter [54]. The phishing filter uses heuristics from the web page content and metadata along with online feedback from a Microsoft server in order to classify sites as safe, possibly unsafe or definitely unsafe.

When some of the heuristics trigger the phishing filter into labelling a site as 'suspicious' a small yellow indicator is displayed in the web browser which, when selected by the user, explains that the site may be problematic and to avoid entering personal information. The URL that the user is visiting is also submitted to the Microsoft server and checked against a white list and a black list.

If the site matches the black list the page is blocked and the user shown a page which explains that this is a phishing web site and recommends they do not use it. They are still given the option to proceed. If they opt out a small indicator still shows that the site may be dangerous.

There is a white list to allow legitimate sites not to be listed as suspicious by registering themselves with Microsoft. Microsoft solicits feedback from users as to which sites are phishing and which are safe which are then checked by employees before being added to the lists.

As described in Section 3.1.2, heuristic classifiers work badly against an adaptive adversary who can test attacks against the classifier and refine them before release. Tuning heuristics can also be very difficult, in particular adjusting the false-positive and -negative rate. If the false-negative rate is too high then phishing web sites are let through unmarked. However, if the false-positive rate is too high users will get used to ignoring any warnings since they often happen on legitimate sites.

[1] `http://www.microsoft.com/windows/products/winfamily/ie/`

To combat this, Microsoft have an online server with black and white lists. The problem with black lists is that they rely on someone reporting the web site before it can be blocked. While the black list distribution system built into Microsoft Phishing Filter will reduce the window in which attacks can happen, there will still be time between the site becoming live and it reaching the black list. At the moment confirmed phishing sites are reported to the relevant service provider and rapidly taken down, but this does not stop the attacks being effective, particularly those mentioned in Section 3.1.4.

The white list mechanism could also be abused. If an attacker can get their site white listed then the content could be changed to be malicious while still being passed by the Phishing Filter. Such white listing would be revoked as soon as it was noticed, but there would be a window for attacks.

Finally, privacy advocates may be suspicious of a technology which submits logs of web access to a server owned by Microsoft.

3.2.2 Firefox phishing protection

Mozilla Firefox[2] 2 contains built in anti-phishing measures [55]. This takes the form of a black list which is regularly synchronized with the Google Safe Browsing black list [56]. In the default mode all phishing checks are done locally by comparing the URL against the synchronized black list. There is also an active mode whereby all URLs are sent to a third party server, such as Google, for verification.

The remarks about black lists in Section 3.2.1 also apply to the Firefox phishing protection. There is a window of the time to be added to the black list plus the delay in synchronizing it (thirty minutes according to the Google terms of use) in which to perform the attack. In addition, the black list is sent out as MD5 [57] hashes of the URL, which means it only matches exactly. This has lead to exploits [58] using tricks where different URLs point to the same page. These variations can be automated in generating spam email and it's not until all URLs have been added to the black list that it is safe.

Again, the online verification has privacy issues.

Firefox 3 contains a URL highlighting scheme to help prevent phishing [59]. Instead of the normal method of displaying the contents of the URL bar all in black, instead Firefox 3 will display only the domain in black and the rest of the URL in a lighter shade of grey.

This has the potential to counter some of the tricks phishers use to mask the real domain, but it does rely on the user realizing the importance of the domain,

[2] http://www.mozilla.com/firefox/

which few are likely to do. There also seems to be an issue with the phisher creating a URL which is longer than the visible bar, thus hiding the highlighted part. Many users are likely not to notice its absence. The misleading domain is also only one trick available to phishers. Relying on the domain as an indicator of trustworthiness fails in the presence of banks who use multiple domains for legitimate services, which they do. In these cases the user can no longer use the fact that the domain is unexpected to class a site as phishing.

3.2.3 Opera fraud protection

Opera[3], the third of the major browsers, also has built in phishing protection [60]. In this case it is in partnership with PhishTank[4], a community black list project. PhishTank takes user submissions of phishing sites and when several people confirm it to be a phishing site add it to a black list. The status of a suspect web site can be checked and the number of votes 'for' or 'against' it being fraudulent checked.

All the comments about black lists apply and the same vulnerability has been found as in Firefox [61]. In this case, however, Opera released a fix for the specific exploit in a later version.

In order to decide how effective community-run evaluation sites were, PhishTank was evaluated by Moore and Clayton [62]. They reviewed the participation rates of the contributors and found that a large percentage of the votes were submitted by a small number of users. This makes it quite open to manipulation and as such should be treated with care. They also compared it to a classic commercial service and found that it performed badly in comparison in both response time and accuracy.

For a number of years Opera has included a feature to prompt when connecting to URLs containing a user name. This is a common trick for obscuring URLs where the user name appears to be the URL and later in the address the real URL occurs. When connecting to such URLs Opera prompts the user displaying the real address and user name. This is an improvement to the new Firefox URL highlighting feature above, but only a slight one.

3.3 Third-party software

Phishing is a sufficiently large problem that there have been many attempts to solve it, many of them as third-party add-ons to web browsers.

[3] `http://www.opera.com/`
[4] `http://www.phishtank.com/`

Several of the pieces of software in this section are toolbars which display some sort of security indicator to assist users in making security decisions, or display an assessment of whether the site can be trusted. Wu et al. [63] performed a study covering these methods which established that even in the presence of such indicators, many users are still taken in by fraudulent web sites.

3.3.1 eBay toolbar

eBay Toolbar [64] is mainly aimed at preventing phishing of eBay and PayPal, the two main eBay brands. These are both targets of a lot of phishing attacks, due to their market share. The eBay Toolbar displays a small icon in the web browser which has three states. When accessing eBay or PayPal it shows green. When accessing a web site in its black list of known phishing sites it shows red. The other protection it has is tracking the user's eBay password and warning if the same password is submitted to other sites.

eBay Toolbar is let down on a few fronts. Firstly, compared to the other black list systems it is likely to have a less comprehensive black list, mainly specific to eBay and PayPal spoofs. The latter will be closely monitored and updated by eBay staff, but updates of other phishing sites is likely to be left to users of eBay Toolbar to report. The response to a black listed site is also suboptimal. As mentioned in [46], security indicators are frequently ignored by users, who are not likely to pay attention to the colour of the indicator when distracted or in a hurry.

The password warning is a more interesting feature. On the one hand it removes from the user the task of assessing whether a site actually belongs to eBay. On the other hand, users are trained to ignore warning dialogs and in many cases will share their eBay password with other sites despite warnings to the contrary.

eBay Toolbar has some interesting features, but is very eBay-specific.

3.3.2 McAfee SiteAdvisor

McAfee SiteAdvisor [65] is another reputation-based toolbar system which assigns ratings to sites as they are visited. In this case, there are three ratings: Safe, Caution and Warning, plus an unknown rating. Whenever the user visits a web page the address is sent to the McAfee's servers and checked against their database. The database is updated by McAfee staff reviewing sites after being alerted by user submissions.

Aside from the problems with all black list systems and the privacy issues detailed in Sections 3.2.2 and 3.2.3, there are also reports of SiteAdvisor incor-

rectly listing sites despite reports being submitted clearly showing the site to be fraudulent [66].

3.3.3 TrustBar

TrustBar [67] is a combination of a black list and some enhancements to TLS.

The black list is similar to all of the others, there is an option to report sites which are vetted by the TrustBar authors and then added to the black list. Since this is a black list specific to TrustBar, as with eBay Toolbar it will evolve more slowly than other lists.

The TLS enhancements try and fix some of the visibility issues with TLS. Details from the certificate are displayed more prominently. TrustBar lists the certificated name of the site and the authority issuing the certificate. In addition, users can select images with which they wish to identify a site, which TrustBar displays on subsequent times the user visits the page. Lastly, TrustBar also keeps a list of well-known sites with both clear and TLS-protected log-in pages and will automatically redirect from the insecure log-in page to the TLS-secured one.

TrustBar provides some much needed additions to online security, mainly those which it could be argued should have been there in the first place. Enforcing use of TLS is something everyone should do, but as seen in Section 3.1.1 this only provides a small increase in security. Making TLS details more visible is an improvement, but as discussed previously, the user cannot distinguish between phishing sites and sites outsourced from the bank. As with the other software solutions listed here this does not protect against a compromised computer.

3.3.4 SpoofStick

SpoofStick[5] installs a toolbar in the browser which prominently displays the root of the domain. This is designed to foil the phishing attack where the real URL is disguised by tricks such as including another domain in the user part of the URL.

SpoofStick is an improvement on the Firefox 3 URL highlighting due to the prominence with which it displays the domain. Other than that it shares the same problems listed in Section 3.2.2.

SpoofStick does not protect against any other type of attack.

3.3.5 SpoofGuard

SpoofGuard [68] uses a comprehensive heuristic-based approach to detecting phishing. Like anti-spam solutions, SpoofGuard uses a range of heuristics to

[5] `http://www.spoofstick.com/`

assign a 'spoof score' to the page and when it passes a certain threshold takes some action.

SpoofGuard has a large range of heuristics. There are several static checks of the web page which is being visited. The URL is checked to see if it contains any of the common methods of obscuring the real address. Images within the page are checked against a database of logos from sites which are often subject to phishing to see if they are appearing on a page not related to the company which owns the logo. Images are compared using a hash algorithm tolerant to small changes in the image. Links within the page are checked as with the URL of the page. Pages which request passwords trigger a check for a valid TLS certificate.

There are also checks which examine the browser's state. The domain of the page is checked against the history to see if it is similar (has a small hamming distance) to other domains which the user has visited. This is to catch typo-squatting phishing described in Section S.Phish. SpoofGuard checks the referrer and assigns a higher score to pages which have been linked from a known web mail site, such as Hotmail. The image check above which initially uses a static database is augmented from pages with logos in the user's history.

The last type of check which SpoofGuard performs is when submitting POST data. Before the data is sent out, SpoofGuard performs a number of checks on the data. If the spoof score is high enough SpoofGuard prevents the information from being sent. SpoofGuard maintains a database of domain, user and password hash tuples. When a form is submitted, SpoofGuard hashes the data and checks it against the database to see if the same password is being submitted to a different site. Other checks also score higher when data is being sent to the site.

SpoofGuard is certainly the most comprehensive of the solutions based on heuristics and prevents the majority of phishing attacks currently seen in the wild. Some of the checks are made less useful by companies who outsource parts of their web presence to other domains and it can be legitimate to include the logo from other companies (for example those with whom they partner).

The other area which uses heuristics is spam prevention. Once heuristic-based spam prevention became popular, the senders of spam quickly adapted such that their messages produced low scores in the classifiers and passed the checks. Since SpoofGuard and the other phishing classifiers are accessible by the attackers there seems no reason why that would not also happen here.

Most of SpoofGuard's checks fail in the presence of more complicated attacks, such as pharming (see Section S.Pharm) and it can be completely bypassed by keyloggers or a compromised computer.

3.3.6 YURL

YURL [69], produced by Waterken Inc, uses 'pet names', an extension of bookmarks to improve the security of Internet banking. The idea is similar to the reasoning in Section 3.1.3: each 'important' web site is labelled with a 'pet name' by the user. This pet name is linked to the hash of the TLS certificate in use. The user then (theoretically) always use that pet name to access and identify that web site. Since the pet name is locally assigned, phishers cannot know what the correct name is to use. The linking with TLS certificates means that the pet name will only be displayed when talking to the correct server, even if a pharming attack has taken place.

The implementation of pet names provided by Waterken does not display the pet name very prominently, making it likely that the user will not notice the lack of a pet name. The correct site lacking a pet name will also occur if the site changes its TLS certificate. This is not something which is guaranteed to remain constant.

The assertion that users will pick a pet name which the attacker cannot guess is also somewhat dubious. When picking a pet name for Barclays, there are a small number of pet names likely to be selected by the majority of the population. Waterken argue that the remote site cannot alter the contents of the pet name toolbar; however, it has been shown [70] that many elements of the web browser can be successfully spoofed in modern browsers.

Lastly, YURL does not protect against compromised terminals and does not protect a user who accesses his bank account from previously unconfigured computers.

3.3.7 SRD

Ye and Smith have a system called "Synchronized Random Dynamic Boundaries" [70] (SRD). This is a technique designed to make it easier for the user to verify that a valid TLS certificate has been presented and harder for an attacker to spoof the indicator.

The core of the scheme is an unpredictable element which is synchronized to a master window elsewhere; Ye uses the border of the window. Because the choice of border is random and constantly changing, an attacker cannot create a spoof indicator, since they cannot know what it should look like. For verification by the user, another window on the machine also displays the correct state, synchronized with all the borders. This approach is easy for humans to verify, although is does clutter the interface and the constant changing can be distracting.

The security which is gained with this system is, unfortunately, quite limited. The presence of a TLS connection and certificate does not in itself guarantee that the user is connected to the server to which they intended to connect. TLS certificates can be obtained very inexpensively for any domain with limited checking done by the issuer. Ye's system does not give any way for the user to check to whom the certificate has been issued. SRD provides no protection against a compromised end station.

3.3.8 DSS

Dhamija and Tygar have proposed a system called Dynamic Security Skins [71] (DSS). This combines the personalization present in YURL pet names and the synchronization from SRD with a trusted input window to create quite a convincing anti-phishing solution.

The idea behind DSS is that the user only enters their credentials into an input window provided by DSS, not directly into the web page. When installing DSS, the user selects a 'skin' for the input window. This is a custom image which is always displayed behind the user name and password fields in the input window which makes it hard for an attacker to spoof the input window, since they cannot replicate the image.

The input window also displays a dynamically generated pattern (called a visual hash) which has been created in cooperation with the server and is mirrored on the input form to which the input dialog is currently attached. In DSS authentication is done not by sending a password to the server, but using the SRP [72] protocol. SRP is a form of Zero-Knowledge proof where the server can authenticate the user without a password being sent over the network. The result of the SRP authentication is a hash which is only known to the server holding the verifier and the client with the password. This hash is used to generate a visual hash which is displayed both on the page and in the trusted input box.

Because the password is not sent to the server a middle-person attack cannot learn the password or know what images to display on a spoofed web page such that they synchronize with the trusted input window.

DSS is not designed to protect against a compromised end station or one with any sort of keystroke logger.

3.3.9 PwdHash

Ross et al. have formalized a protocol which several people have speculated over in PwdHash [73]. PwdHash is designed to deal with the problem of users sharing

credentials between multiple sites and also with the problem of users submitting passwords to the wrong site.

It is implemented with a browser plug-in which identifies password elements in HTML forms and transparently rewrites them on form submission. The content of the password element is rewritten to be a hash of the content, the user name and the host name to which the form is being submitted.

This ensures that stealing the password database of a site or sniffing the password while the user is logging in does not reveal the secret needed to log into other sites, even if the user thinks they have the same password. In addition, if the user is ever tricked into entering their password into a malicious site, it will not be sent the same password as the real site if the host names do not match.

In order to support operation with a browser which does not have the PwdHash plug-in (for example in an Internet café), there is also an online method of generating the hash. This is a web site which takes password, user name and host and turns this into a hash which may be copied into the log-in form. There are also some issues with initialization and sites with multiple host names using the same authentication database which are to some extent handled in the paper.

For very limited security goals, PwdHash works well. It definitely mitigates some of the risk from users who aren't very security-aware and defeats some of the more simple phishing attacks. More complicated attacks such as pharming will still work (the host name matches and is sent the correct hash) as will any attack which compromises the user's terminal or keyboard; capturing the password out-of-band is sufficient to generate the hashes.

3.4 Bank-provided measures

The banks have also not been idle about preventing phishing attacks.

3.4.1 TANs

Transaction Authorization Numbers [74, pp.13] (TANs) are a form of one-time passwords used by some banks. The customer is issued with a sheet containing multiple passwords, typically this is done via some out-of-band mechanism, usually the postal service or in person at a bank branch. For each transaction the next TAN in order is requested in order to complete the transaction.

TANs are either essentially the same as Lamport hash chains [75] used by authentication systems like One-time Passwords In Everything [76] or more simply just a list of random numbers which are stored on the server. As with other one-time password authentication systems TANs protect against snooping of the

credentials on a legitimate transaction in order to use them later. However, TANs are no defence against a middle-person attack in which the victim is convinced to connect to the attacker instead of the real bank and the transaction challenges and responses are forwarded between the real bank and the user, but the transaction details are altered to transfer funds to the attacker instead. There have also been cases of phishing web sites harvesting several codes for later use, without the complexity of a middle-person attack [77].

3.4.2 SecurID

RSA sell a selection of two-factor authentication tokens under the SecurID [78] brand. Several other suppliers have similar offerings, such as VeriSign [79]. They are all based on the same principle of a time-synchronized code encrypted under a key shared between the token and the authentication server.

SecurID 200/600/700

The SecurID 200 has a credit card form factor and SecurID 600/700 are key-fob designs. The three are otherwise functionally equivalent. They provide a six digit digital display which refreshes every minute. In the current implementation each code is the AES-hash [80] of the current time and the device-specific key. Previous versions have used modified RC2 [81]. To log into a server the user must present their password and the current hash value. The authenticating server has the matching key for the token and a synchronized clock so that it can calculate the same hash.

The time-dependent part of the credentials is designed to prevent copying of the credentials and reply attacks; the physical token prevents the user giving away their password. The protection is actually limited to a window of sixty seconds. If the attackers can get a log-in session before the code expires by eavesdropping the log-in process then they can use that session (although the code won't be good for future sessions). This can be mitigated somewhat by allowing only one simultaneous log-in or only one log-in per minute. These strategies are not without their drawbacks and they still fail to protect against the more serious threats outlined in Chapter 2.

The use of these time-dependent codes have the same middle-person vulnerability as TANs and one which has already been seen in the wild [82]. If an attacker can convince the user to connect to their web site instead of the real one, either through phishing, pharming or Trojans, the user will enter the credentials and the session can be use to carry out other transactions without the knowledge of the user.

Finally, if the attacker can steal the token as well as the password they can use it without further hindrance to generate the codes and log in themselves.

SecurID 520

The SecurID 520 is similar to the SecurID 200 except that it also includes a PIN pad. The user must enter their PIN before a code is displayed. The resulting code is then a function of the PIN, the code and the time.

This is additional protection against theft of the token, since stealing the PIN is significantly more difficult than stealing the token or the password. It does not improve any of the other security features, in particular there is no extra protection against the middle-person attack.

SecurID 800

The SecurID 800 is a modification to the SecurID 700 key-fob token. It also includes a universal serial bus (USB) interface which allows applications to programmatically request authentication codes from the device without the user having to transcribe numbers. This is good for convenience, but actually bad for security. The account cannot be accessed without the presence of the token, but if the terminal to which it is connected has been compromised, a malicious program can request any number of authentication codes without the user being aware.

The SecurID 800 also includes functions for doing other digital certificate signing and management. These are typically not used for authentication.

SecurID 990

The SecurID 990 is another credit card form factor device which in addition to the authentication code in the SecurID 200 also provides challenge and response functionality via a 10-digit PIN pad.

The normal SecurID code is used to log in, but when specific transactions are requested a challenge is presented in the form of a confirmation number which they must enter into the token. The token signs this and displays a response which the user types back into the site to authorize the transaction.

This reduces the available window of attack: a middle-person attack cannot make arbitrary numbers of transactions once they have access to a session. However, it does not stop a middle-person attack entirely. The attacker can only perform the same number of transactions as the user is trying to make, but any of those transactions can be rewritten at will by the attacker. The challenge code presented to the user gives no indication as to the nature of the transaction and

therefore could be authorizing any transaction, not necessarily the one the user intended to authorize.

SecurID summary

The RSA SecurID range raises the bar slightly for attacks on online transactions. However, none of the solutions prevent middle-person attacks, which are in current use. The USB variants fail particularly badly in the presence of a compromised terminal.

3.4.3 CAP

One form of multi-factor authentication which banks are starting to introduce derives from the EMV (Chip and PIN) specification [83] and is called chip authentication protocol [84]. This is currently being rolled out in the UK by Barclays [85].

The CAP reader is a handheld device very similar to a calculator. It has a slot in the top for inserting a credit or debit card, a PIN pad and a small display on the front. It uses the feature of the EMV protocol whereby the card will produce the message authentication code (MAC) of data with which it is supplied using the secret key it shares with the bank. There are two primary modes of operation with CAP, generating random nonces and challenge/response.

Random nonce generation

The first mode, which in current proposals will be used for most types of transaction, uses the card to generate a random nonce. When performing a transaction the web site will ask the user to generate a code with the reader which they do by inserting their debit card and pressing the 'new code' button. The CAP reader displays the code on the screen and the user transcribes this into the web site and the server checks the MAC is valid.

This scheme has many of the flaws present in other schemes presented above. In particular, it is essentially a weaker form of SecurID. It is susceptible to real-time and off-line middle-person attacks because the user cannot know what they are authorizing. In addition, the code is not time-dependent, so can be cached by the attacker for later use.

Challenge-Response

The second mode is a challenge-response mode. Current proposals use this mode for transactions such as setting up new mandates. In this case the user is asked to enter some details from the transaction as part of the challenge. These are then

sent to the card which produces the MAC under the key shared with the bank. The user copies the number into the web site as before.

This is an improvement over many of the schemes discussed in this chapter, but unless the user has to enter all the details of the transaction there is still scope for a middle-person attack. In addition, since it is only used for setting up mandates, there are many types of transaction which are still vulnerable.

Anderson [86] has made similar comments about the Chip Authentication Protocol.

3.4.4 SMS challenges

Another approach which has been used is to use the short message service (SMS) as a secondary secure channel [87]. In this system a challenge is sent over SMS which the user must enter into their computer.

Mobile phones look like a good platform for a secondary secure channel. They are small, portable and ubiquitous. However, the continuing convergence of functions on to mobile devices has resulted in mobile phones being essentially general purpose computing devices and suffering from the same vulnerabilities as personal computers. Several viruses, worms and Trojans have been produced for Symbian [88, 89] and WinCE [90] (the two most popular smart phone operating systems) as well as an exploit for the iPhone [91]. While some of these are just proofs of concept, history has shown that such things quickly turn into real outbreaks, particularly when there is money to be made. If mobile phones become key in financially-sensitive transactions they will quickly become a target for virus writers and those who pay them. As mobile phones continue to accrue functionality, the complexity (and therefore scope for security-sensitive bugs) increases massively.

All this suggests the conclusion that mobile phones are not a long term solution for securing financial transactions. A real attack was seen on the Ubuntu Foundation [92] in which the SIM card belonging to the foundation's chief financial officer was reissued to the attacker and the challenge sent by the bank intercepted. In addition, sites like FakeMyText.com[6] also do not encourage trust in the SMS network. One bank will even send challenges via email [93]. Finally, the recent scandal of wiretapped Greek mobile phones [94], while a very sophisticated attack which probably used an insider, proves that attacks on mobile phone networks are possible. This was also studied in a report in 2002 for the Mobile Payment Forum [95] which analyzed the impact and feasibility of a number of attacks on payment systems using mobile phones including viruses, message in-

[6] http://www.fakemytext.com/

terception and injection. It concludes with best practices to mitigate these attacks, but not eliminate them.

One of the other inevitable results of mobile phone convergence is that they will in some cases be the primary computer which is being used to perform transactions. This increases the risk of a compromise since the second authentication system is no longer a separate device.

3.4.5 On-screen keyboard

In response to losses from keyboard sniffing programs, Citibank UK has introduced PIN entry with the mouse via an on-screen keyboard [96]. This is billed as foiling keystroke loggers since no keys are pressed. This is similar in principle to the drop-down boxes that other banks such as Barclays use on their Internet banking sites.

These techniques do not, however, protect against Trojan horses which have for years had the ability to capture screenshots and mouse movements. This was demonstrated against Citibank by Yash [97].

The on-screen keyboard also has several other problems. It does not allow mixed-case passwords, thus reducing the space from which people can select passwords; it makes 'shoulder surfing' attacks easier and it stops visually impaired people from accessing the site.

3.5 Other multi-factor systems

The literature also contains proposals for multi-factor authentication systems to combat phishing, mostly involving mobile devices.

3.5.1 Two-factor mobile authentication

Pietro, Gianluigi and Strangio [98] devised a protocol which uses the short-range wireless Bluetooth protocol to interface a token (in the form of a smart phone) to a user's computer. Hundreds of millions of smart phones have been sold to date and they provide a convenient portable platform which many consumers already have.

The protocol suggested in this paper involves three parties. The bank, the user's computer and the smart phone. When the user registers, the bank generates authentication details which the user's computer transfers to the device. When logging in, the agent on the user's computer and the smart phone mutually authenticate each other using the key which the user entered into the agent

on the computer. Once this has been accomplished, the computer allows a connection to the bank.

This scheme is resistant against a number of common attacks, but it is not clear that active middle-person is protected against, nor is compromise of either the mobile device or the terminal.

3.5.2 Phoolproof phishing prevention

Parno, Kuo and Perrig [99] present the alliteratively named "Phoolproof Phishing Prevention". This scheme offloads part of the TLS verification to the mobile computing device. The user accesses their online banking by selecting a bookmark on their mobile phone which knows the server's URL TLS certificate. The mobile directs the browser to connect to the URL and verifies the certificate it presents. The TLS client authentication is performed using a client certificate which is stored on the mobile which assists in creating the TLS channel.

This approach successfully stops the standard middle-person approach and at first glance is better than simply using TLS client certificates in the browser combined with better certificate checking. However, since the TLS session once setup is entirely accessible by the client, a compromised machine could perform a middle-person attack on the content of the session, replacing transaction requests with others and masking the replies. It also seems possible to create a second TLS connection in the background connecting to the bank and injecting arbitrary content while the browser actually connects to the attacker.

3.5.3 Cronto

Cronto[7] are designing a selection of tokens, either based on software for mobile phones or a dedicated token. These tokens use either a camera built into the mobile phone or the token to read what Cronto call a 'visual cryptogram' [100] containing details of the transaction and an authentication code to be copied back to the bank.

As with CAP the presence of part of the transaction details is a big advantage in security over most of the other systems presented here. There are still a number of limitations, however. The solutions without a dedicated device are still vulnerable to either a compromised computer or mobile phone. The implementation of a dedicated device does not display many details of the transaction, leaving some avenues of attack.

[7] http://www.cronto.com/

3.6 Defence effectiveness

In Chapter 2 attack vectors on e-banking were described using an attack tree. This can also be used to analyse how effective various defences are against those attacks.

There are several classes of defence which will be highlighted below using the attack tree notation.

3.6.1 TLS

Figure 3.1 is the attack tree with the attacks that TLS stops highlighted. As can be seen it does not affect many of the attack routes, in particular those nodes which are involved in either the traditional phishing attack, or the more complex attacks which are starting to appear.

The public-key infrastructure (PKI) around TLS is designed to stop rather more. All of the attacks which involve spoofing real web sites, for example. However, it has been shown time and time again that this is not effective. The security indicators and prompts provided by TLS implementations are almost entirely ignored by the users.

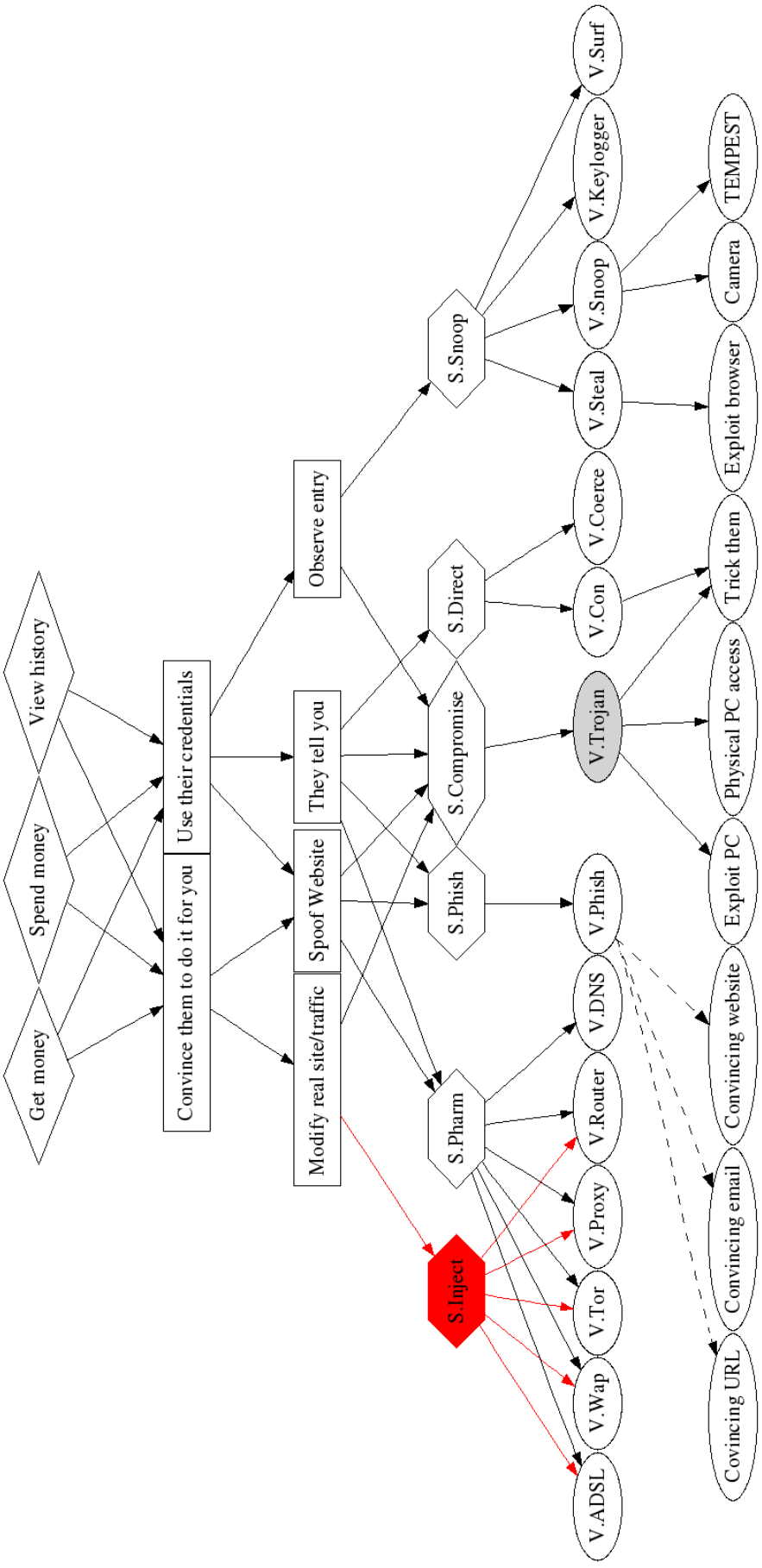

Figure 3.1: Online banking attack tree with TLS

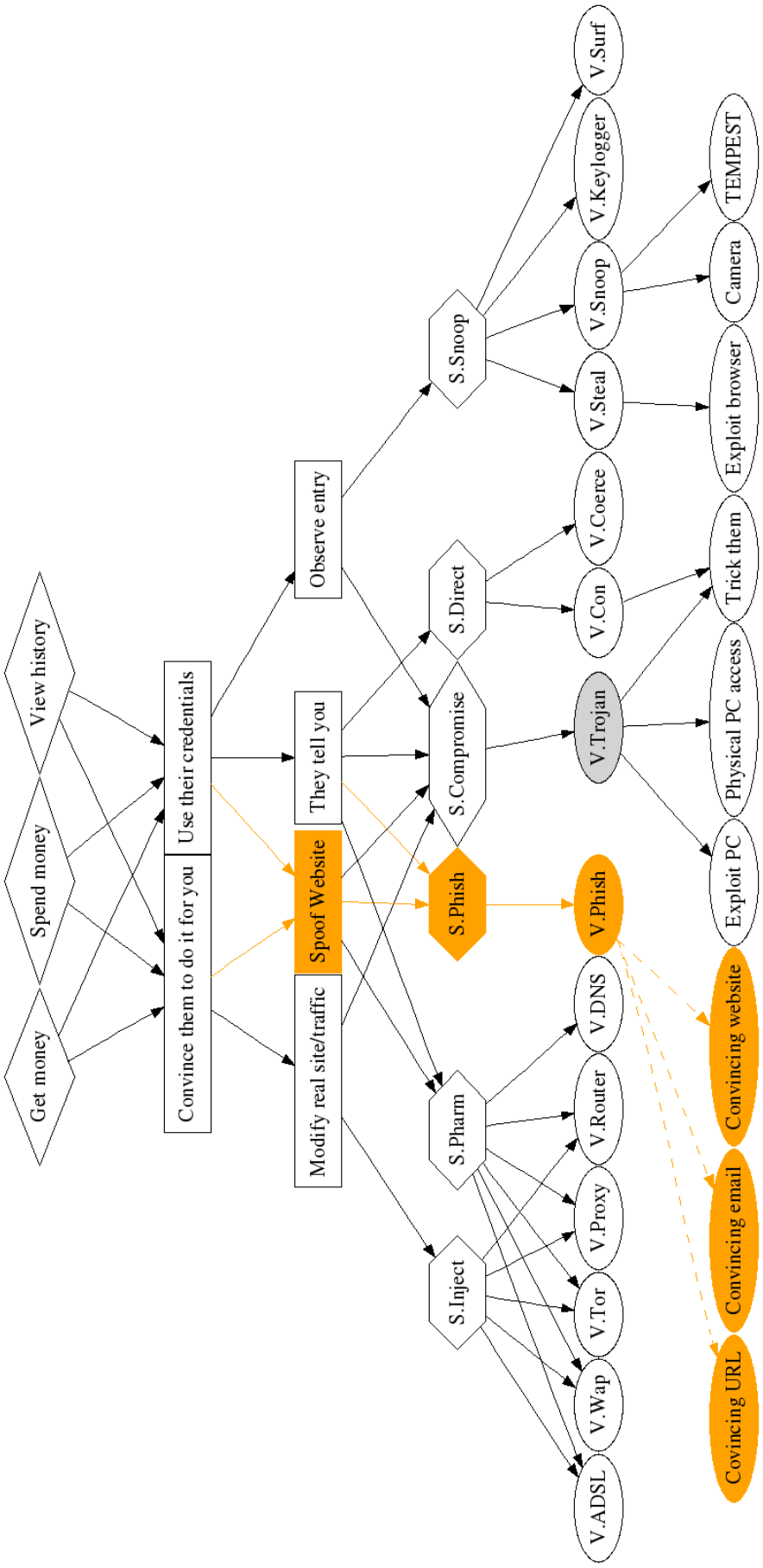

Figure 3.2: Online banking attack tree with detection of phishing attempts

3.6.2 Phish-detection

The phishing site detection software, such as is built into the major web browsers, is represented on the attack tree in Figure 3.2. Here the defence is given in orange to show it is not complete protection. Heuristic detection will never match 100% (and the references in Chapter 3 suggest it is significantly less than that) and black lists by definition have a certain amount of lag before being updated. Tactics from the Rockphish gang also foil the majority of black list-based solutions around.

3.6.3 Software enhancements

The three other software-based enhancements, SRD, DSS and PwdHash have been compiled on to Figure 3.3. Again there is a focus on protecting credentials from snooping and preventing phishing attacks from working. Unlike heuristic detection they should stop the attacks shown all the time, but it is obvious there are still many routes through the tree.

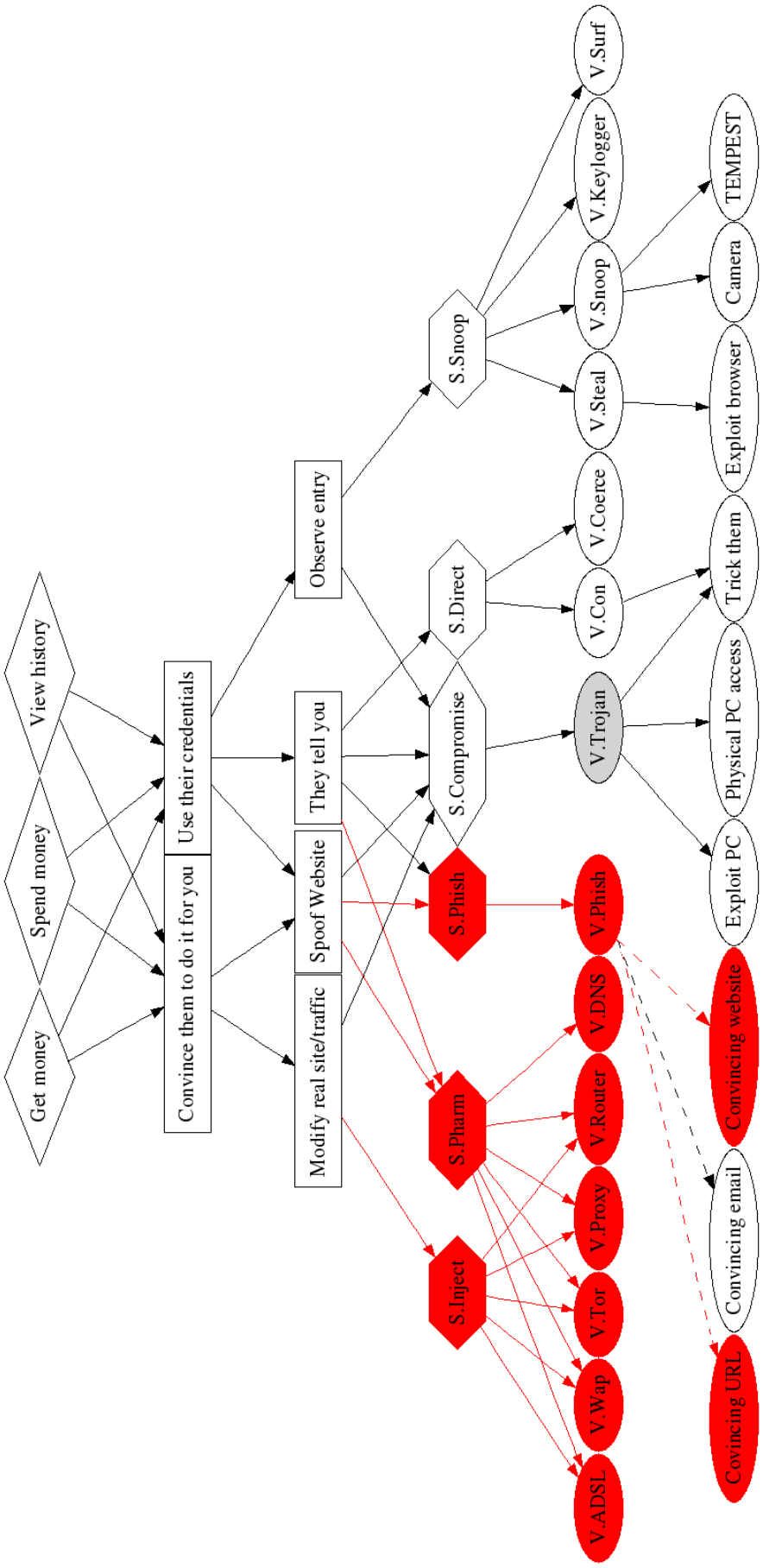

Figure 3.3: Online banking attack tree with extra software-based defences

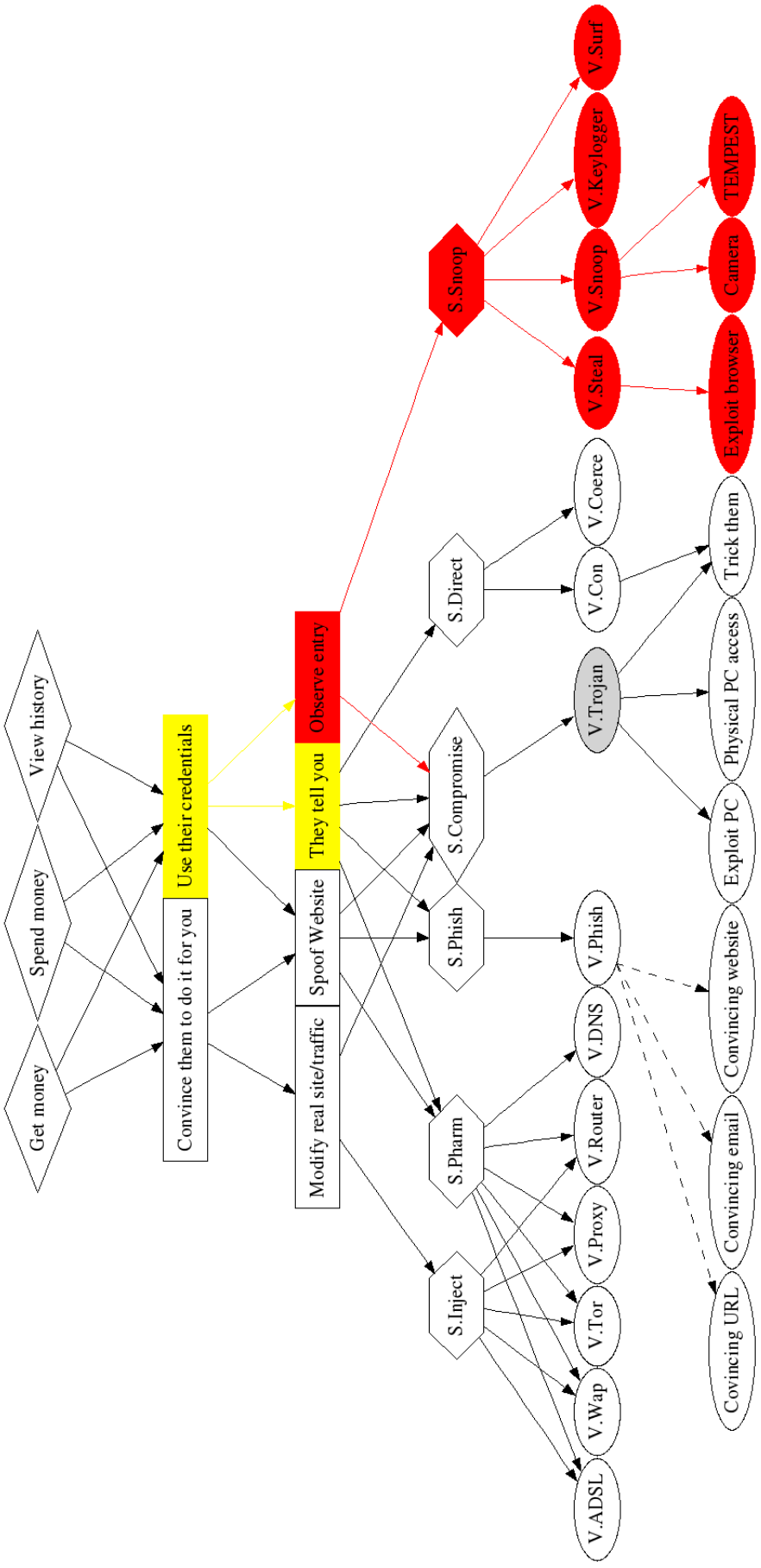

Figure 3.4: Online banking attack tree with tokens

3.6.4 Tokens

There are two types of tokens mentioned here which will be classified as dynamic and static. Static tokens include TANs and dynamic ones SecurID. Figure 3.4 has two colours. Dynamic tokens prevent all routes marked in red and yellow. Static tokens only prevent the attacks in red. They are still susceptible to the list of codes being revealed to the attacker in whole or in part.

3.6.5 Summary

As can be seen from the graphs in this chapter none of the reviewed defences do anything about the problem of a compromised terminal. The 'trojan' node which was highlighted on the first graph in Chapter 2 is not touched on any of the graphs in this chapter. This means that none of these defences are effective against an attack involving compromise of the user's computer.

Chapter 4

Introducing the banking dongle

In the previous two chapters it has been shown that the common, well-known attacks on Internet banking are not the only attacks which criminals are capable of performing. It was also shown that the majority of the protections currently afforded to Internet banking and similar targets are aimed at stopping the common attacks and not more sophisticated attacks. Even those which claim to be more thorough do not successfully prevent the exploits. The particular attack to which there is no robust defence is that of compromise of the computer which is being used to access the online banking service.

The Trojan is the most powerful attack vector available to attackers and the most difficult to defend against. It is no wonder, therefore, that many systems for protecting online banking have chosen to conveniently ignore this in their attacker model. For a while this was an acceptable amount of risk, because that class of attack was relatively uncommon. Even now it is not wide-spread, but the instances of it which have been seen, suggest that as less complicated attacks become less successful there will be no problem in the attacker switching to using Trojans.

What is required is a defence which is robust even in the presence of a Trojan on the computer which is being used to access the service.

This chapter outlines what is required in such a defence and then describes a possible solution. Much of this work was presented at the The 12th Nordic Workshop on Secure IT-systems held at the University of Reykjavik in October 2007 [1].

4.1 Transaction transparency

In 2001 Anderson wrote about the chosen protocol attack [101, pp.24–25]. This has particular relevance because the example he used, if only in the abstract, was with online shopping. The 'Mafia in the middle' attack, as it was also dubbed, is

a variation on the middle-person attack. The attacker places himself between the victim and the target of the attack. The classic middle-person attack involves the attacker somehow spoofing the target so that the victim connects to the attack by mistake.

The chosen protocol attack is, however, much more general than that. In the Mafia example given by Anderson, the Mafia puts up a web site dealing in pornography and entices the victims to access it. This can be entirely legal, if of dubious morals in some people's views. When accessing the pornography, the victim is asked to prove their age by signing some random value using their credit card. This has none of the visual clues of a spoofed banking site; the customer expects to be doing this and is on the site they wish to be. However, in parallel to this, the Mafia are conducting a high-value transaction elsewhere. The "random value" which the victim is asked to sign is actually the authentication for this high-value transaction.

A chosen protocol attack boils down to an authorization which the user thinks is for one purpose actually being used for a different purpose. Chapter 3 describes several systems in use by banks today which are vulnerable to such an attack.

Solutions for this involve ensuring that messages are unambiguous as to their purpose and destination and not allowing third parties to use the same authentication system. The problem with this is that there is a growing desire from consumers and merchants for a single system which they can all use. In addition, the unambiguity of the message must be clear to the user, not just to the computers involved. This poses a problem when the assumption is that the computer in use is compromised.

This lack of transparency is at the root of all phishing and pharming attacks. If the user could be sure they knew what they were authorizing these frauds would not be possible.

4.2 Low-cost device

One of the key premises of this research is that the result could be a consumer device. This requires it to be producible cheaply and in bulk.

The canonical example of a cheap device held by consumers is the smart card. These are sufficiently cheap that banks can issue them to all customers. Producing something this cheap would be extremely difficult; however, this may not be necessary.

There are two approaches for increasing the price point at which such a device is viable. Firstly is billing it as a 'platinum' option for their high-value customers.

Those customers are often more aware of the issues and would value a bank which offered such a device.

Secondly it could be offered as an after-market third-party offering which the consumer purchases. If enough benefit can be provided to the consumer it would be a viable proposition. See Chapter 5 for some ideas on how to make it attractive for the consumer.

Such a USB-attached device with a display is comparable in complexity and therefore price, to USB Internet phones which typically retail at around £10–£30 [102, 103]. Alternatively, Crystalfontz[1] sell a number of USB-attached liquid crystal displays with keypads for £15–£30.

Oikonomakos, Fournier and Moore have developed cryptographic hardware in polysilicon [104]. While currently just proof-of-concept, the ability to produce significant logic along with TFT driver circuitry should provide a simple solution with few components for the device which has been proposed. This will be sufficiently cheap to produce in bulk that it will be attractive to both banks and consumers.

4.3 Form factor

There are several form factors which could be used for this device. Initially this device was dubbed a banking 'dongle' due to the envisaged form factor of a small, USB-attached device with a display and one or two buttons. There may be other form factors and connectivity which could be used for the same result.

4.3.1 USB

USB is now the standard interconnect for computer peripherals and is supported in all major operating systems. This results in it being a cheap, easy to use and ubiquitous solution, which is very suitable for the needs of this sort of device.

4.3.2 Bluetooth

Alternative form factors are possible. Bluetooth is the current short-range wireless protocol of choice and would provide a suitable connection to a host computer.

[1] http://www.crystalfontz.com/

4.3.3 2-D barcodes

An alternative which is popular in Japan is the 2-D barcode [105]. These can be read by a camera on a mobile phone and therefore have the advantage of not needing any special hardware support on the client (USB is almost ubiquitous, but is disabled in some workplaces for security policy reasons). A version of this is being developed by Cronto [106] for exactly this use in their security products.

4.4 Device IO

The device presents a trusted user interface to the customer. This requires a screen to display details of the transactions and some method of authorizing or denying transactions.

The minimal implementation of this is a pair of 'OK' and 'Cancel' buttons. Since performing transactions also requires logging into the bank's web site with traditional credentials this is already a two-factor authentication scheme; but this can be improved upon with the trade-off being increasing the cost of the device and a small amount of extra interaction with the device for the user.

One option would be to include a PIN pad and require entering a PIN to confirm a transaction. Another option would be to use biometrics, probably in the form of a fingerprint scanner in the device. Both of these would reduce the exposure from a stolen device but are orthogonal to the rest of the ideas presented here.

4.5 Protocols

The transport will forward the protocol messages from the device to the bank's server. The protocol for each transaction can be quite simple, although there are many issues with key setup which are not directly addressed here. Some ideas are presented in Chapter 6

The protocols given here assume that a good block cipher is in use (AES [107] is a suitable candidate, but the protocols are by no means limited to that) in a suitable mode (including, but not limited to, cipher-block chaining (CBC) [108] with a random initialization vector (IV)). The message authentication code used is also not specified but assumed to be good. Cipher-based MAC (CMAC) [109] is a candidate for this. They also assume that there is already a shared key between the device and the bank. This is practical in the case where the device is provided by the bank to its customers.

4.5.1 Cipher choice

The protocols below are presented with implementations using particular symmetric and asymmetric ciphers. These have been chosen for simplicity, easy of implementation in hardware or existing use in hardware which would be appropriate to use. Some cipher-specific issues (mainly due to block and key length) are discussed but essentially they can be replaced in the abstract protocol diagrams with any cipher of that type. As Schneier [110] states, AES is probably the symmetric cipher of choice for the majority of applications.

The abstract protocol notation used in this chapter does not assume a specific block cipher; however, Appendix A which gives message details is implemented using Triple-DES due to the limitations of the device which was used as an example. Changing this will only vary the block sizes and padding in the message details.

4.5.2 Protocol definition

$$
\begin{array}{rll}
M_1 = & I, \text{“INIT”}, Len, D, K_{BD1}, K_{BD2} & \\
B \rightarrow D: & \text{“INIT”}, Len, IV, \{M_1\}_{K_{LT}}, MAC_{K_{LTM}}(M_1) & (1) \\
M_2 = & I+1, \text{“ACK”}, Len, D & \\
D \rightarrow B: & \text{“ACK”}, Len, IV, \{M_2\}_{K_{BD1}}, MAC_{K_{BD_2}}(M_2) & (2) \\
M_3 = & N, \text{“TRANS”}, Len, D, \text{transaction}, Type & \\
B \rightarrow D: & \text{“TRANS”}, Len, IV, \{M_3\}_{K_{BD1}}, MAC_{K_{BD2}}(M_3) & (3) \\
D \rightarrow U: & \text{transaction} & (4) \\
U \rightarrow D: & Auth & (5) \\
M_6 = & N+1, \text{“AUTH”}, Len, D, \text{transaction}, Type, Auth & \\
D \rightarrow B: & \text{“AUTH”}, Len, IV, \{M_6\}_{K_{BD1}}, MAC_{K_{BD2}}(M_6) & (6)
\end{array}
$$

Figure 4.1: Banking dongle transaction protocol

Figure 4.1 shows the protocol. The notation used is fairly standard. Messages are given as source → destination and then a list of the message fields. Encryption is denoted by braces subscripted by the key used for the encryption. MACs are denoted by MAC subscripted by the key and then parentheses containing the data to be used to calculate the MAC.

The principals involved in the protocol are the bank (B), the device (D) and the user (U).

Message (1) in this protocol is the session-key initialization message. It is encrypted under the long term shared key (K_{LT}) and with a MAC using the long

term shared MAC key (K_{LTM}). It is sent at the start of each session and contains the message type and length (Len), two block cipher keys (K_{BD1}, K_{BD2}), the name of the destination device (D) and an incrementing counter (I) identifying this run of the protocol and the keys used to prevent replay attacks. These keys are used for encryption and MAC respectively in the rest of the protocol. This message is acknowledged by message (2) which confirms the receipt of the keys by encrypting $I + 1$ under the session keys.

Messages (3) and (6) perform a transaction. These messages contain the message type and length, a block encrypted under the session key, a MAC of the encrypted block under the session MAC key and the IV for both.

The transaction request message (3) contains in the encrypted block a copy of the plaintext data plus the destination ID to prevent attacks where a valid encrypted block is sent with different data in the plain-text part of the method. The rest of the ciphertext contains the length of the transaction data, data describing the transaction taking place and the type of transaction. There is also a nonce (N) to provide freshness guarantees, which is unique to this run of the protocol between the device and the bank. The description of the transaction will include any information necessary to describe the transaction. Typically this will be the amount, identifiers for the source and destination and the unit of currency.

Between the last two messages in the protocol between the device and the bank the user is shown the transaction details on the trusted user interface. They then have to make an authorization decision on the transaction and either confirm or deny the transaction using a button on the device.

The response (6) repeats all the information from the request including the nonce incremented by one and also inputs an authorization code to indicate whether the transaction should proceed. Again there is a MAC to ensure integrity of the message.

4.5.3 Security analysis

Here follows an informal security analysis of the above protocol.

Attacker model

It is assumed that the attacker has complete control over the end user's computer and all communications links between the bank and the user. Thus it is possible to observe and modify any messages as well as performing more high-level attacks such as DNS spoofing and the sending of fake messages.

Passive attacks

The first type of attacks are passive attacks. These are ones in which the attacker merely observes the messages. They are the easiest to protect against, and hence the least useful. The goal of an attacker in this scenario is to recover the key in order to send fake messages or to infer some information about the messages.

The messages are all encrypted in cipher-block chaining mode with a random initial value and a nonce or ID in each message to ensure that all messages are different and no inference can be drawn from identical repeated messages or parts of messages. The cipher chosen should prevent any other attacks; however, to prevent too much ciphertext being generated under the same key transactions are encrypted under a session key which is regenerated each session.

Active attacks

Active attacks cover every situation in which the attacker alters the message flow. This includes inserting and deleting messages and modifying or replaying existing messages.

Messages are protected by a message authentication code with a different key from the encryption which is also generated for each session. The MAC provides integrity protection against modifying the message.

Message insertion and replay attacks are the main threat against the protocol. All the messages include their message type and destination along with a unique value for that run of the protocol. This prevents messages being used in different runs of the protocol from that intended.

Replaying an entire protocol run using an old initialization message is prevented by the inclusion of an incrementing counter for each run of the protocol. The bank keeps track of these and aborts connections with repeated counters.

Duplicate transaction request messages may be sent and will be accepted by the device, however they will not correspond to an outstanding transaction when a reply is received by the bank and will have no effect. This is similarly true of repeating the transaction response messages.

Finally there is the attack of deleting messages. An attacker can simply perform a denial of service attack by dropping some or all of the messages in the protocol. In most cases it will be obvious to the user that a denial of service is happening. However, if the attacker drops the last message in the protocol and then forges a 'transaction complete' message on the computer they may believe the transaction has completed. A possible solution to this would be to add in a seventh message which displays the transaction confirmation on the device. However, it is not obvious whether such an attack would be of any gain.

The key initialization acknowledgement message ensures that deleting the session-key initialization message is noticed early in the protocol run by the bank. Stopping new session keys being used by the device cannot cause more ciphertext to be sent under the same key since it will only respond to a valid message from the bank (which will not be sent). Generating an extra response to an existing message through replay with a different authorization from the user is possible, but since a random IV is chosen nothing can be inferred from the two messages. Deleting any of the initialization, acknowledgement, transaction request or response messages will merely cause the protocol run to fail.

Cipher attacks

There are several different classes of cryptanalysis on ciphers. Some of these are only possible when a cipher is used in certain modes in the given application. The most basic attack (and the least useful) is the known ciphertext attack, which is always present. The use of the cryptographic primitives in these protocols prevent several more powerful attacks being performed on the underlying cipher.

The known plaintext attack involves the attacker having corresponding plaintext and ciphertext. Portions of the encrypted messages are predictable (the type and size are also sent in plaintext with the message) and an attacker who has compromised the host will be able to infer the transaction content. However, each message is sent with a different nonce and IV. The CBC mode which is employed for the cipher ensures that plaintext differences at the start of the message are spread throughout the ciphertext and the known plaintext is of no use.

The chosen plaintext attack is one in which the attacker can cause certain plaintexts of their choice to be encrypted under the key. To a small extent this is possible; if the host has been compromised then the attacker may initiate transactions choosing some of the transaction data. The nonce, IV and CBC mode as before make it difficult to get any information from the chosen plaintext since it is only part of the plaintext.

The plaintext is never revealed from any ciphertext so chosen ciphertext attacks are not possible.

4.5.4 Use of the protocol

An important part of protocols is defining how they should be used; many good protocols are broken because they are used badly in practice. Therefore, the following additional restrictions are placed on use of the protocol.

- Session Key Lifetime: Key initialization should be performed each time the client applet connects to the bank.

- Failure Handling: All messages whose MAC fails to validate, the protected data doesn't match the public data or the destination doesn't match the receiver's ID should be dropped without acknowledgement.
- Key ID handling: Session key IDs must only ever increase.
- Replay Protection: Banks should discard all messages other than transaction responses with a valid MAC, matching encrypted and plaintext data and a nonce which corresponds to an outstanding transaction request.
- Transaction Overload: A transaction should either be authorized, declined or timeout before displaying another request. New transactions should not be accepted within a given time of the previous request.

The protocol is designed to be used in a system where the end station is compromised. Therefore, transport layer security between the server and the applet, such as TLS, does not help. Any attacks which could be made on an unsecured transport can also be made on the end station after decryption. However, since the technology is already in place there is no penalty for using it and it adds an extra layer of defence against some classes of attack. Defence in depth is a good security principle since it allows for failures in parts of the system without this compromising the whole system.

With that in mind there are several other recommendations which can be made regarding the use of these protocols.

- The connection between the applet and the server and the web browser and the server should be TLS encrypted with a certificate signed by a trusted certification authority.
- The applet should be given a key in parameters from the web page which is passed to the server with requests.

These recommendations prevent most of the attacks possible without compromising the user's terminal including in particular denial of service attacks which would otherwise be possible on this protocol. This denial of service is still possible with a compromised end station; but that is always going to be the case.

4.5.5 Analysis

The defences covered in Chapter 3 were analysed in the attack tree notation from Chapter 2. With this analysis it can be seen how much better the system proposed here fares. Figure 4.2 shows the attack tree with all routes prevented by the banking dongle. As can be seen, almost all the graph is coloured red. Notable

exceptions to this are the nodes involving the victim deliberately giving the attacker money. If the attacker can either by force or deception get the victim to pay them in full knowledge that this is what they are doing, then there is not much any system can do to stop this. It is even debatable whether this is a failing of the system.

There are two other things worthy of mention which aren't covered by this graph. Firstly, those routes which end in viewing the victim's transaction history rather than getting their money. The banking dongle is not designed to prevent these attacks. Since all the interaction with the bank other than confirming transactions is performed through the regular web browser then all the attacks on web browser systems can still view and modify this. It is only the performing of transactions which is restricted as shown by Figure 4.2.

The other important thing is to note that whenever a defence system is added this inevitably changes the attack tree. Importantly in this case there is a new avenue of attack involving theft of the dongle. Physical theft of a token is an attack which does not scale well, but for high-value targets is still viable.

Token theft can be mitigated through a variety of techniques. Using the other techniques given in Chapter 3 in cooperation with the banking dongle requires that the attacker defeat both systems in order to defraud the victim. An attacker who is targeting a victim sufficiently to be able to steal their dongle has a good chance of also installing a keyboard logger or Trojan on their computer. As has been shown, this is sufficient to bypass other types of defence.

A more effective technique would be to include some sort of access control on the device itself as discussed in Section 4.4.

4.6 Usability Issues

The acceptance of any solution by the users will depend heavily on the usability of the system. The device presented in this chapter has been designed with usability and the security of the interface in mind. To that end the interface is designed to be very simple. In the limit, all that is required are two buttons and a screen. One interaction per transaction is the minimum which is possible if the user is to authenticate every transaction (which Chapters 2 and 3 show is necessary) and that is all this device requires

In addition Section 4.4 shows that the banking dongle can be provided in a form factor which requires little or no software setup on the computer with which it is used. All of the interactions with the bank are automatic and transparent.

There are some issues with initial setup and personalization of the device itself, associating it with the bank and specific user, which impact the usability.

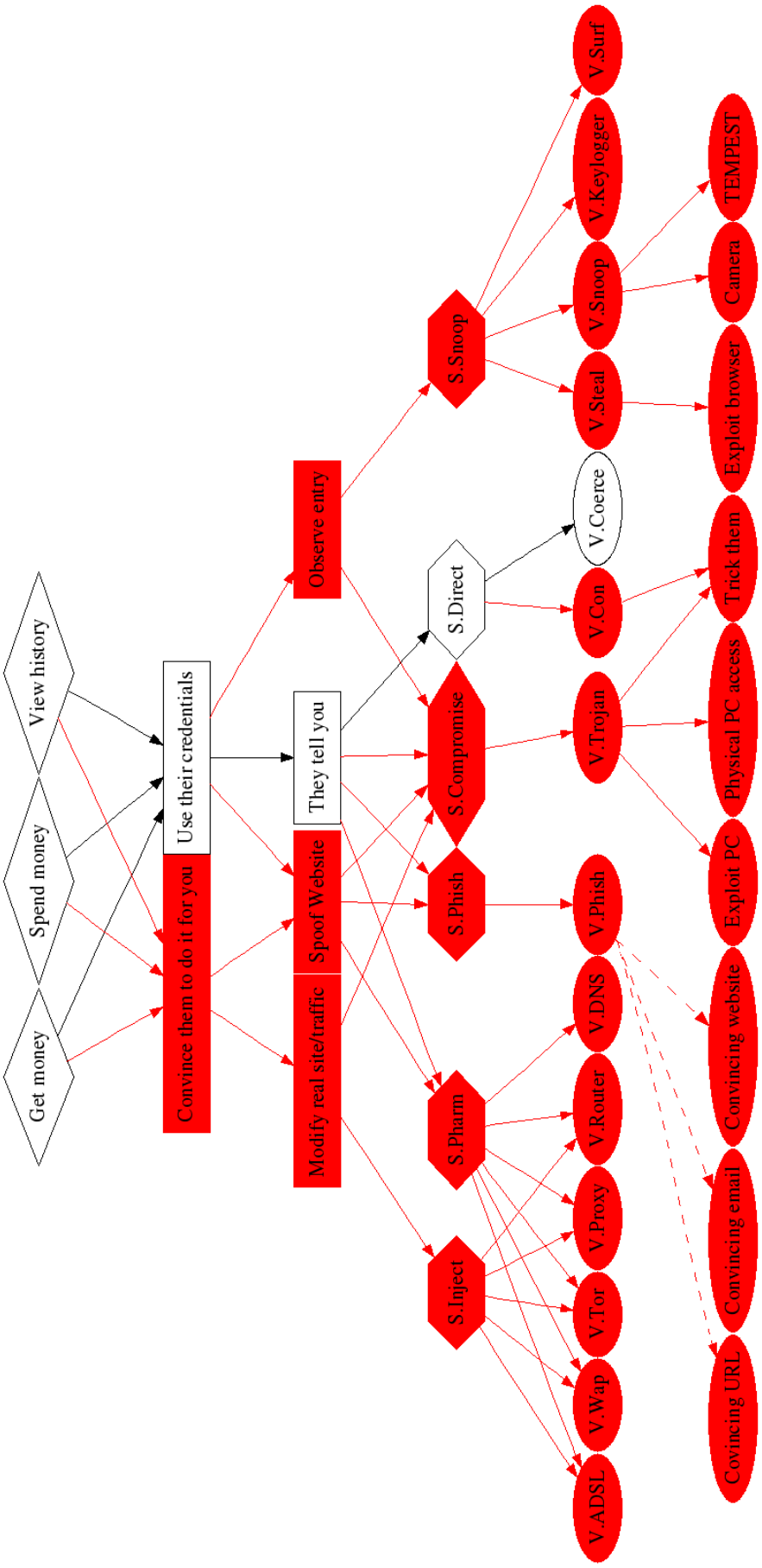

Figure 4.2: Online banking attack tree for the banking dongle

These topics are covered in Section 6.3.

4.7 Demonstration system

To investigate how the protocol described in Section 4.5.2 would work in practice a demonstration system has been designed and built. Since no actual devices have been produced this can only be a proof-of-concept demonstration. In this system the device is modelled by a laptop running a Java program.

4.7.1 Structure

The structure of the demonstration system can be seen in Figure 4.3. There are three main components, each of which comprises several sub-components.

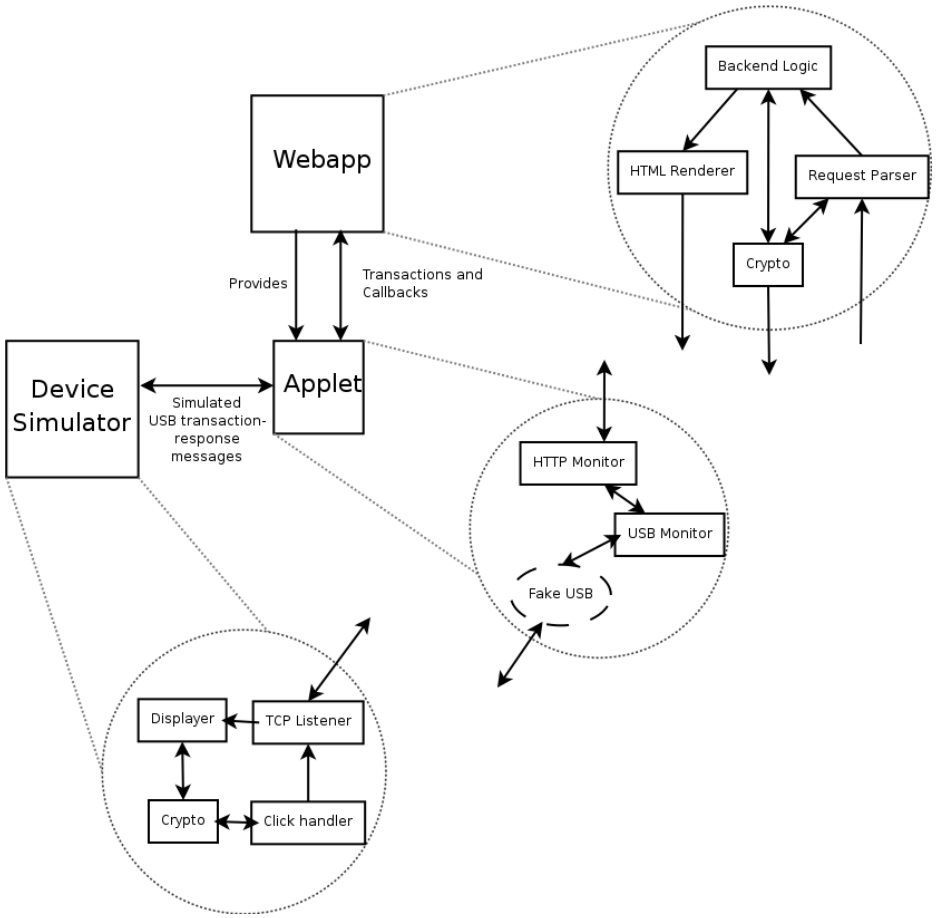

Figure 4.3: Demo system structure

4.7.2 Device

The secure display device is simulated using a laptop running a Java applet with separate cryptography and display modules. This obviously does not have

the desired security properties of the secure display, but will demonstrate the protocol-level of the system. A screenshot of the demo running is given in Figure 4.4.

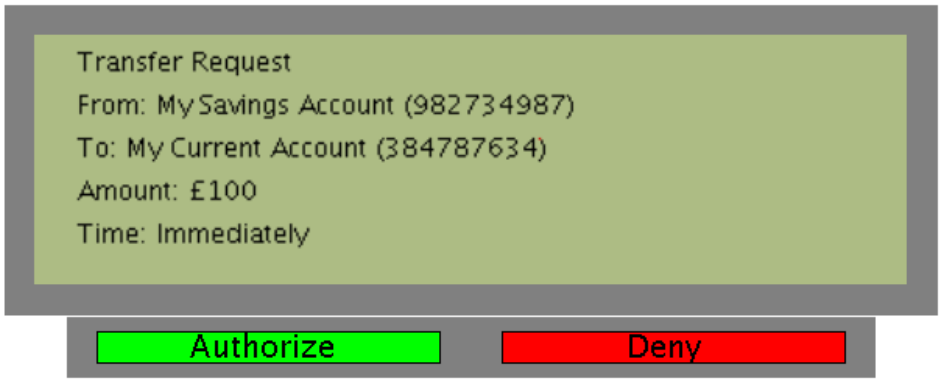

Figure 4.4: Screenshot of secure device prototype

4.7.3 Bank

The online banking web site is being modelled by a locally-hosted web site running the demonstration software. Several common transaction types are allowed on simulated accounts. These are all verified using the protocol outlined above.

When the web site is viewed a second page is opened containing the applet. Cookies are used to make sure there is always exactly one copy of this page open while the web site is being browsed. A second connection is opened by the applet to the web server on which transaction requests are made. When the user requests a transaction using the web site, an encrypted transaction block is sent to the applet and the site waits for a valid response before carrying out the operation.

The transactions available are transferring money between accounts and paying money to an account at another bank.

4.7.4 Applet

The applet connects the device and the web server together. This part of the demo system is essentially the same as would be used in a real deployment. Written in Java, the applet will work on all the main-stream browsers on all the major operating systems without the installation of any software which wouldn't be installed on an average computer. This includes those in Internet Cafés and Kiosks.

The applet polls the web server for new transaction requests which are then written to a USB pipe connected to the secure device. Call back handlers from

the USB sub-system are used to get transaction replies and then submit them to the web server using HTTP POST requests.

The USB sub-system in the demo system is actually a wrapper around a TCP socket which is connected to the laptop running the device demo program. This fake sub-system implements the same application-programming interface (API) as the real Java USB sub-system so this can be directly swapped out for use with a real device.

Appendix B gives more detail on the implementation of the applet.

4.7.5 Demo conclusions

The demonstration system presented above provides an example of how some parts of a complete system would be implemented. It also provides a prototype of possible device user interface so that UI issues can be considered ahead of developing a real device. The user interface presented here shows that all of the necessary information to verify a transaction can be clearly displayed on a small amount of screen real estate.

The main result from this demonstration is to confirm the use of a Java applet as a zero-install method of communication with the device. This was implemented using the Java USB API but it could equally well be used to display a 2-D or Cronto barcode as suggested in Section 4.3.3.

4.8 Summary

This chapter presents the idea of a separate secure device which would create a secure channel between the bank and the user. The device would have a sufficiently small trusted computing base that it can be analyzed for security issues. Protocols are given which use such a device connected to the untrusted computer to provide enough security to authorize transactions while the bulk of the interaction is still with the primary computer.

Almost all of the attacks in Chapter 2 are prevented through use of these protocols and such a device. This shows that a relatively inexpensive device can provide robust security against the majority of attacks which are possible today, not just those which are in wide-spread use.

Chapter 5

Protecting the consumer

WHEN A CONSUMER FALLS victim to a fraudulent transaction there are in theory a number of safeguards to ensure they are not left out of pocket. The Banking Code [111] and the fraud protection offered by most credit card issuers are examples of these. However, in some cases these are not sufficient.

5.1 The balance of power

In today's payment systems the banks have a lot of power over both the merchant and the consumer. The threat of government regulation has, at least in the UK, resulted in banks forming a collaborative code of practice which has held off any official oversight. However, this has resulted in the banking industry being essentially self-regulated. Even the various ombudsmen and dispute-resolution services are appointed by the banks.

One can hope that competition between banks and concern for their reputations would result in them erring on the side of benefiting their customers, but in some cases even this is not enough.

5.1.1 Legal history

There have been a number of cases in the UK and in other countries where victims of fraud have not been compensated by their bank as a matter of course, despite claims in the banking code otherwise.

Researchers at the University of Cambridge are often involved with disputes between banks and consumers and in the case of phantom withdrawals have compiled a web site [1] listing many cases where the bank initially refused compensation. A phantom withdrawal is a cash withdrawal from a cash point in which the consumer has not been negligent with their PIN and yet cash has still

[1] `http://www.phantomwithdrawals.com/`

been removed. While some of the cases were eventually resolved successfully a number are still outstanding and in all cases the bank tried to shift the blame and liability on to the consumer. This has also been the case in a number of other countries [112, 113].

The same research group has also published a web site dedicated to chip and PIN[2] which describes how chip and PIN relates to the customer, particularly in the areas of liability and dispute resolution. These findings have been condensed into a white paper by Anderson et al. [114]. While the banking code says that disputed amounts should be refunded unless the bank can show that the customer acted fraudulently or without reasonable care; in practice it seems this decision is left to individual bank investigators and there is no standard of proof required for this decision. Once they decide not to reimburse there is little a consumer can do. In fact, it appears chip and PIN is being used as an excuse to claim the banking system is now infallible and therefore the customer must have been negligent with their PIN [115].

Some doubt has also been cast over the process of the Financial Ombudsman Service [116]. Anderson claims that the adjudicators employed by the ombudsman made numerous errors both of law and of technology which suggests the rulings they came to should be treated with some suspicion. Notes on specific cases which support Anderson's claims have been submitted to the Hunt review by the Foundation for Information Policy Research [117].

5.2 Electronic attorneys

While proposing the 'Electronic Attorney' [118], Anderson and Bond make the observation that in the current banking system there are devices which protect the interests of the bank and devices which protect the interests of the merchant but that these interests do not always align with those of the consumer. The 'Electronic Attorney' would be used in combination with a credit card in order to protect the interests of the consumer.

If consumer has a device which takes part in the transaction process then it can also protect the consumer's interests.

5.3 Audit logs

When consumers are the victims of fraud the banks are meant to refund their money. However, when the transaction has been authorized by PIN, banks claim that it is the responsibility of the customer. Drimer and Murdoch [119] have

[2] http://www.chipandspin.co.uk/

shown a number of attacks which produce PIN-authorized transactions and the banks have been unwilling to provide evidence that the MACs generated by the card to authorize the transactions are valid. There are also suggestions that this might be applied to Internet transactions via the SecureCode/Verified by Visa schemes described in Section 6.4.1.

Therefore, it would help if the consumers could be provided with an audit trail which they could show to prove they had not made the transaction. This would level the playing field between the consumers and the banks. To that end, the device proposed in Chapter 4 should produce an audit log which will verify that the contents are both complete and accurate.

5.3.1 Log storage

To avoid increasing the secure storage requirements on the device the audit messages once created form a continuous trail which can be stored on a untrusted medium and can be verified as long as it is unaltered. This allows the device to be sold as a third-party service in which the only trust relationship necessary is that the consumer trusts the provider to keep an accurate record of the log. Once produced anyone verifying the log does not need to trust the provider.

5.3.2 Log creation

Figure 5.1 shows a version of the protocol from Chapter 4 adapted to produce a log. This is similar to the protocol presented by Schneier [120], however, this protocol does not consider the bank to be trusted to look after the consumer's interest, only its own. Integrating the logging with the transaction protocol results in the bank being part of the authentication of the log. This is discussed in more detail in Section 5.3.4.

The same notation is used as in Figure 4.1 on page 57. The principals are the bank (B), device (D) and the audit log (L). Log entries are denoted by L_I and the nonce for that entry as O_I. Nonces denoted by O_I are used as commitments between log entries and form part of the hash chain within the log; they are completely independent to the nonce N used in the protocol run between the bank and the device. The new cryptographic primitive in use here is a hash function, denoted by $h()$. This should be a secure hash function such as SHA-256 [121] or preferably the result of the current NIST competition [122] to establish a new secure hash standard, which will be published as SHA-3.

Messages (1) through (5) are the same as those in Figure 4.1 and have therefore been omitted for brevity. In message (6), the device includes in the transaction authorization a hash of the previous transaction and of the nonce for the next

$$
\begin{aligned}
M_6 = \quad & N+1, \text{“AUTH”}, Len, D, h(O_{I+1}), h(L_{I-1}), \\
& \text{transaction}, Type, Auth, MAC_{K_D}(\text{transaction}) \\
D \rightarrow B: \quad & \text{“AUTH”}, Len, IV, \{M_6\}_{K_{BD1}}, MAC_{K_{BD2}}(M_6) \qquad (6) \\
M_7 = \quad & N+1, \text{“TACK”}, Len, D, MAC_{K_B}(N, I, \text{“TACK”}, Len, D, \\
& \text{transaction}, h(O_{I+1}), h(L_{I-1}), Type, Auth, MAC_{K_D}(\text{transaction})) \\
B \rightarrow D: \quad & \text{“TACK”}, Len, IV, \{M_7\}_{K_{BD1}}, MAC_{K_{BD2}}(M_7) \qquad (7) \\
L_I = \quad & N, I, D, Len, Type, \text{transaction}, M_7, \\
& h(O_{I+1}), h(L_{I-1}), O_I, Auth \\
D \rightarrow L: \quad & L_I, MAC_{K_D}(L_I) \qquad (8)
\end{aligned}
$$

Figure 5.1: Audit protocol

transaction. Message (7) is the bank acknowledging the transaction and in the process validating the completeness of the log.

The final message is the log entry which is stored. It contains the transaction, the bank's confirmation of the transaction as well as a hash/nonce chain linking them with the rest of the log. The whole log entry is signed by the device.

In contrast to the protocol in Section 4.5.2 the bank is not trusted. The messages are all encrypted under a key shared with a bank, but there are also additional message authentication codes computed with a key which is only know by the device and the device manufacturer (K_D). The bank also does not trust the customer, so there are also message authentication codes computed with a key only known by the bank (K_B).

User interface

From the point of view of the user, creating the log entries can be completely transparent. The user interacts with the device in exactly the same fashion but it also generates log entries which are sent wherever they need to be stored.

It may be desirable for some of this process to be visible to the user so that they can confirm that everything is working. Part of the log entry could be displayed on the device after the transaction, such as the MAC over the entry. It could also be displayed on the user's computer. Because it is not security critical, but merely to assure the user that things are working correctly, it does not matter if it is suppressed when something goes wrong. On the other hand it is always better not to encourage the users to rely on data which is not trusted.

5.3.3 Verifying the log

In the case of a dispute arising, the customer would turn over the device and the logs to an impartial third party, such as the police. There are then two possible next steps. If the disputed transaction is less recent than the end of the log, the log file is complete with no further action.

If the disputed transaction is more recent, or if there is any dispute about more recent transactions, then another transaction is performed in the presence of the third party and a representative of both parties. Assuming the transaction succeeds, checking the cryptographic chain in the log is enough to prove that the log is complete.

If the bank wishes to disagree with any part of the log, they have to produce the key which created the MACs in the logs and show that it created a non-disputed transaction (possibly the one done in the controlled environment), but not the ones they dispute.

5.3.4 Security analysis

The log is designed to be presented to an impartial third party who should be able to verify that it is both accurate and complete. In particular, the customer is producing the log in order to refute a transaction which the bank claims they made. Since the customer is the one relying on the log it should be the customer, or an agent they trust, who stores it not the bank. However, there is no security requirement on the storage other than that the customer trusts it to store the log entries accurately. In particular, the verifier needs to place no trust in the party storing the logs.

The security properties which are desirable, therefore, are that the consumer cannot forge a log which doesn't contain a real transaction while still appearing valid and that the bank cannot assert that it is not complete when it is.

Looking at these in turn, to create a log which omits a transaction there are several attacks which may be tried. Firstly, can an entry be removed from the log once created. Each entry contains the hash of the previous entry and the hash of a nonce which appears in the next one. If there are any subsequent transactions in the log it will be apparent if an entry has been removed since the chain of hashes will be broken. In addition, the nonce revealed in the later entry won't match the hash earlier committed to.

This leads on to asking whether the next log entry can be massaged so that the missing entry is not noticed. The revealed nonce and the hash of the last transaction could be swapped, but those values are included in the MAC of the log entry. They are also included in the MAC which the bank provides when

confirming the transaction. To fake these MACs the attacker would need the secret keys of both the bank and the device.

The next attack is to try and replace a log entry with a mundane one. Again, the next entry in the log would have a hash which did not match and would need to be forged as above. A more interesting attack is to see whether by manipulating the protocol during the protocol run it is possible to immediately create a transaction which follows the last-but-one entry without including the last entry. If it were possible to run the protocol with the bank giving the same nonce-hash and last-transaction hash on a subsequent message then a MAC could be obtained which fits in the chain. However, this would require access to the shared key between the device and the bank. Even in that case, the bank would notice the repeated hashes and be able to produce a duplicate message with a MAC created by the device.

Finally, there are attacks by the bank. The bank takes part in the log protocol by producing a MAC on the hashes forming the log chain in each message. To claim that the customer performed a transaction which is not in the log, they would have to be able to point to the place in the log in which it took place. However, the MAC the bank produced is stored in the log as part of the transaction authorization and certifies the entries either side of it. To claim that the log was not complete they would have to claim that the whole log was fabricated, which would require denying that the customer performed the other transactions.

Chapter 6

Variations on the banking dongle

CHAPTER 4 DESCRIBES a specific solution to the e-banking problem which uses bidirectional secure communication with a dedicated device. There may also be solutions which use hardware with other properties which could solve the problem.

6.1 Unidirectional security

The protocol in Chapter 4 uses a bidirectional secure channel. It might be useful to speculate whether a similar level of security could be achieved if the authentic channel were only in one direction. There are a number of reason why this might be useful. Firstly, it may reduce the cost of the secure device. Even small reduction in the individual cost of a device represents vast savings to a bank trying to roll this out to many customers. Secondly, there is some existing technology which could provide a similar unidirectional secure channel relying on a small trusted computing base. If it is possible to secure transactions using a unidirectional protocol it may be possible to use such technology rather than introducing an additional device.

A number of content producers are trying to produce digital restrictions management solutions which are resistant to the user accessing the content in digital form. High-bandwidth digital content protection (HDCP) is a cryptographic protocol for encrypting content between consumer devices, including graphics cards and monitors. The aim is for software running on the computer not to be able to read the data being displayed on the monitor. This is very similar to the e-banking transparency goal that no software running on the computer should be able to alter the data being displayed to the user.

Another solution in a similar vein dubbed 'CryptoGraphics' has been proposed by Cook et al [123, 124]. This proposal is very similar to HDCP but depends on explicit keying. The protocol given by Cook includes secure keying

from a system outside the graphics card, such as a smart card, to provide session keys used for the decryption. This smart card could be provided by the bank and these session keys allow for much more robust encryption protocols than HDCP.

Using a system such as HDCP or CryptoGraphics introduces new potential vulnerabilities, such as authenticating and identifying to the user whether data has been provided over the secure channel or from another source. Many of these security issues have been discussed in the field of multi-level secure systems and some can be eliminated by careful use of the protocols involved.

With only secure communication in a single direction the difficulty is for the user to be able to approve or deny a transaction such that it is unambiguous which transaction is being approved. Cryptographic signatures or MACs are traditionally used for these applications; however, this scenario limits what can be used to what a user can be reasonably expected to calculate or transcribe. This leads to a different level of errors which the system must cope with.

6.2 Unidirectional protocol

The next protocol is based around a device which can only do secure communications in one direction. This means that data can be securely displayed to the user, but there is no secure communication in the other direction.

The protocol is based on the one given in Section 4.5. The secure channel receives an authentication code which is then copied by the user into the normal channel to the bank. The protocol is given in Figure 6.1.

$$
\begin{aligned}
M_1 = &\quad I, \text{“INIT”}, Len, D, K_{BD1}, K_{BD2} \\
M_2 = &\quad N, \text{“TRANS”}, Len, D, \text{transaction}, Type \\
B \rightarrow D: &\quad \text{“TRANS”}, Len, IV, \{M_1\}_{K_{LT}}, MAC_{K_{LTM}}(M_1), \\
&\quad Len, IV, \{M_2\}_{K_{BD1}}, MAC_{K_{BD2}}(M_2) \qquad (1) \\
M_3 = &\quad N+1, \text{“AUTH”}, Len, D, \text{transaction}, Type \\
D \rightarrow U: &\quad \text{transaction}, check(MAC_{K_{BD2}}\{M_3\}) \qquad (2) \\
U \rightarrow B: &\quad check(MAC_{K_{BD2}}\{M_3\}) \qquad (3)
\end{aligned}
$$

Figure 6.1: Unidirectional protocol

The first message in this protocol conflates the session key setup and transaction start steps in the original protocol. Because there is only a unidirectional channel to the device it cannot acknowledge the session keys. Instead, the bank attempts a transaction in the same message, the transaction part encrypted under the session keys sent in the first part.

Key and transaction acknowledgement are performed in the final two steps. Messages 2 and 3 are the transaction response being sent via the user over the low-security channel back to the bank. As discussed in the next section, the $check$ function represents the last four decimal digits of the input with two check digits added to the end.

6.2.1 Transaction response

The transaction response needs to fulfil the difficult criteria of being both easy to accurately transcribe for the user and yet also secure so that an attacker cannot guess it. To match the security of the bidirectional protocol a response of 16, 32 or 64 digits, depending on the security parameters, would have to be used. This is clearly not something a human will want to accurately copy every time they do a transaction.

Since this is not an attempt to replace Chip and PIN with this system, but rather enhance it, then to some extent no more security is required than the PIN which banks already issue to customers. This means having an entropy of at least 10^4, but also not allowing any attacks where many responses can be tested for validity.

Not allowing multiple tries contrasts sharply with the requirement for accurate transcription. People will often make mistakes when transcribing things and declining transactions or locking accounts because of it is at best unfriendly.

There are schemes which can be used to detect errors when numbers are input at the expense of adding a digit or two to the end. These are not cryptographically secure (it is easy to calculate a different, but valid set of numbers) but are designed to catch common typographic errors with high probability. With one of these check digit schemes it should be possible to assume that errors where the check digit does not match are due to human error and errors which have a valid check digit, but the transaction authorization code does not match are due to a malicious attacker.

ISO 7064 [125] standardizes a two-digit scheme based on the modulus base 97. It catches all single substitution, transposition and shift errors along with 99% of double substitution and other errors [125, pp. 12]. This would increase the length of the response code by two digits.

These extra two digits cannot be included in the security margin because the attacker can trivially calculate them. Therefore to match the entropy of PINs at least four digits are needed before adding the check digit or six digits in total.

6.2.2 Restrictions on protocol

Section 4.5.4 gives the restrictions on use of the bidirectional protocol which implementations are expected to follow in practice. The unidirectional protocol follows the same rules but with some additions due to the response codes.

- Invalid check digits: An invalid check digit should merely prompt the user for the code again with no indication of whether the rest of the code was correct.
- Valid but incorrect code: A response code which has a valid check digit but does not match the one sent out should immediately cancel the pending transaction. Optionally it can also put some form of lock on the account.
- Response code reuse: Each new (if otherwise identical) transaction must have a new, unpredictable, response code.
- Transaction response input should always be accompanied by the identifier of the transaction it corresponds to.

These four rules ensure that users should never cause any sort of lock down or cancellation of a transaction they intended to authorize through typographic errors, but that attackers get no more than one try to guess the code for any given transaction.

6.3 Key distribution

One of the largest problems which has not yet touched been upon is that of key establishment. In most of computer security, once there is a securely established key known to be between only the two desired parties the rest is simple. The question is how this key is agreed upon securely.

6.3.1 PKI

The first obvious solution is to use a PKI in the same way that TLS is used with banking web sites. However, this requires a trusted PKI which has additional key distribution and trust issues. In addition to this, simply having the bank authenticated to the device is not enough. An attacker could register their own device to the target's account. So, both ends need to be authenticated.

Client authentication is also supported in most PKIs (including TLS), but it is rarely used. Becoming certified is generally only something which companies want to do once per application. Certifying all the users with the PKI is a large

overhead. It also requires some way of inputting the keys or certificates from the PKI into the device, or extracting the key to send to the PKI infrastructure.

6.3.2 Bank-owned device

The second solution would be to take the device in person to the bank and have it initialized there. This pushes the security and trust problem into the real world domain which banks have much experience in. There are other problems raised by this solution, however. Firstly, there are now several branchless banks and credit card issuers. Secondly, there are still insider attacks. Secure hardware and requiring authorization by several principals mitigate these at the cost of money and time.

6.3.3 Existing shared secrets

One way to make the system require less overhead would be to use some existing shared secret between the parties to bootstrap the system.

Enhanced Diffie-Helman

Several extensions to traditional Diffie-Helman [126] use a small shared secret to generate a strong shared secret while authenticating the two endpoints. This is also used in protocols such as SRP [127]. If the secure device has some way to input a PIN or similar secret then the bank could send a PIN via an out-of-band method. This is already trusted for the PINs for credit cards.

CAP

Alternatively it could be possible to bootstrap the device from an existing trust relationship between the bank and the user; that is, from their credit card. The chip authentication protocol uses the fact that off-line an EMV card will create a MAC for data which it is sent using a key shared with the bank. If the secure device included a card reader this could be used as the input to an enhanced form of Diffie-Helman.

6.3.4 Postal service

The banking industry relies heavily on the security of the postal service. PINs, cards and Internet banking details are all sent in the post. Similarly, the keys for this scheme could be sent via the postal service. This may be a full-length key or a shorter secret for use in an authenticated key exchange protocol such as in Section 6.3.3.

Whatever sort of secret is sent through the post it must be entered by the user directly into the device. This will require some sort of entry system. A keypad is relatively expensive and error prone for the user. A camera to read a 2-D barcode is easier to use.

6.3.5 Multiple accounts

Until now this thesis has assumed that the user of the device will be using it to access a single account at a single bank. Obviously most people have several accounts, often at different banks.

A simple solution to this would be to provide a separate dongle for each account. This is obviously impractical for the user. When considering multiple accounts at the same bank it is already the case that the credentials a user has for online banking allows them to access all of their accounts. Similarly, a device such at this could be used for accessing all the accounts. Adding the source account, preferably with a user-assigned alias, rather than the account number, to the transaction details displayed on the device disambiguates which account is being used for the transaction.

Accounts at several different banks is a more difficult problem. Typically the multi-factor systems today are specific to each bank and a user will have one for each bank with which he has an account. If this device were a third-party device, rather than one supplied by the bank, it would be possible to perform multiple key setup steps, one for each bank. Several possible methods for key setup are given above. As with multiple accounts with a single bank, transaction details can include the name of the bank from which the funds are being transferred.

6.4 Beyond Internet banking

So far the discussion has only been about protecting Internet banking. While this is an important goal, there are many other situations in which financially sensitive transactions take place and ideally these would also be protected.

6.4.1 Online shopping

The first obvious place in which a robust solution to online fraud would be useful is online shopping. Retailers are moving online faster than banks, many only have online outlets and no traditional shops at all.

The discussion of vulnerabilities in Chapter 2 applies equally to online shopping. Many retailers have accounts with saved card details with a simple username and password authentication which is subject to all the same attacks as

online banking credentials. The introduction of CVV2 was designed to prevent attacks through stealing the database from the merchant but does nothing to stop attacks on the consumer. Indeed, in many retailers it is just the card details and no password whatsoever.

The difficulty with introducing a more robust security solution throughout online retailers is that there are many more smaller companies which would have to adopt it, resulting in significant integration costs.

Verified by Visa

There exists a scheme in use by Visa [128] and a similar one by MasterCard [129] where merchant transactions are verified in cooperation with the card issuer. These represent an infrastructure for hooking into the flow of transactions in the retailer's systems. They have been pushed by Visa and MasterCard to traders under conditions of their card processing agreements and allow Visa and MasterCard to request authentication and authorization directly from the customer.

In the current implementation users are redirected to a web site run by the card issuer which requests additional authentication details, at the moment a password. This increases the security of general online transactions to that of online banking. As has been seen though, online banking is not that secure. In addition, they are training users to enter authentication details into web sites which they are redirected to while shopping. This is something users should be discouraged from doing, not encouraged.

While the current implementations of these systems leaves much to be desired they do represent a method of deploying more robust systems which is already in use by merchants. Systems such as the security device proposed in Chapter 4 could be integrated into many retailers' transactions with modifications only made to the card issuers' systems. Since it is the card issuer who is liable for the fraud and whose systems must communicate with the device this has the correct incentives in order to be accepted in practice.

6.4.2 Non-financial systems

There are a number of systems outside the financial realm which could benefit from this sort of robust defence against fraud, the most obvious being more general digital signature schemes.

For the last few years digital signatures of one form or another have had some weight in law in a number of countries [130]. They suffer from the same problems as financial transactions, however. It may be possible to prove that some device holding a key signed a particular bit string, possibly in conjunction with a PIN

only known by the principal associated with that key, but unless there is a trusted path for the bitstring (or preimage of the bitstring) between what the user sees and the signing operation, nothing really can be said about whether that was what the principal intended to sign.

Some new issues come into play when a system is used for more general transactions. Physically the device must be able to display a much more varied and larger amount of information to the user and therefore needs to be correspondingly larger and hence more expensive. There are also issues concerning the format of the bitstring which is signed and whether it is unambiguous as to how it is displayed and what it means. In principle, however, the same system can be used and more trust can be put in any digital signatures generated using it.

Chapter 7

Conclusions

THE UNDERLYING HYPOTHESIS of this work is that banking fraud cannot be eliminated without a dedicated, trusted security device. Stopping common forms of e-banking fraud is not sufficient to protect against the criminals.

Chapter 2 investigated the possible avenues of attack on Internet transactions and showed that the traditional phishing attack is only a small part of the attack landscape. With reference to attacks which have been seen in the wild it was shown that these other avenues of attack are implementable by today's fraudsters.

Chapter 3 categorized a number of current defensive schemes, showing that the reactive model of defence development at best combats specific threats and at worst is only partially successful against those threats. All the defences reviewed are vulnerable to at least one of the attack routes investigated in the previous chapter.

Chapter 4 introduced a more robust scheme for authentication and authorization of online transactions. This works using a trusted device to create a very small trusted computing base, enabling secure communication with a bank without relying on the security of any of the intervening computers. This includes the computer which the customer is using to access the e-banking web site. The device forms a trusted path from the bank to the customer.

The idea of a trusted path is key to this work. Most solutions at best provide a trusted path to the user's computer (many do not even do this), however, general purpose computers are not themselves trustworthy agents of the user's intentions. This has been seen through the many exploits and Trojans, some of which specifically target Internet banking, to which general purpose computers are subject.

7.1 Proposal evaluation

The proposed device negates the problems with a compromised computer by providing a trusted, authenticated path to the user over which all transactions are authorized. Because it is guaranteed that each transaction will have the correct details shown to the user the principle on which all of the attacks on online banking are based is removed. Thus, even the most powerful attack, the Trojan, is prevented.

In addition, because the device is the minimum necessary to provide the desired functionality it is possible to audit it for security vulnerabilities. It is also possible to build it with some amount of tamper resistance and hence protect it against attackers with much larger resources than is normally the case. This means that real assurances can be made that the authorization seen by the bank is the same as the one shown to the user, the only way to stop the whole class of attacks.

As always no solution is a panacea. There are a number of drawbacks to the proposed system. Firstly interoperability. In the past systems like this have failed because of interoperability issues. The proposal tries to mitigate a number of these, helped by the recent standardization of I/O connections and the emergence of portable languages such as Java. Suggestions for alternative methods of communication have also been made which further ameliorate those problems.

Secondly, there are still some avenues of attack left open. Obviously if an attacker can threaten the customer directly, or deceive them sufficiently, the customer may deliberately authorize a transaction to the attacker. There is only so far that technical solutions can go to prevent such abuses and such attacks are outside the scope of this work.

This device also does nothing to keep the transaction log secret. The primary interface for transactions is still the computer, with just transactions being confirmed through the device. Protecting against reading of the transaction log requires all interaction to be done through the trusted path. This would significantly increase the cost and reduce the ability to audit the device.

7.2 Proposal adoption

One of the main stumbling blocks for the adoption of a scheme like this is cost. Unlike smart cards, this sort of system is unlikely to see complete market penetration since it will be an order of magnitude more expensive. Chapter 5 describes a scheme which would provide more incentive for the consumer to spend what is still a reasonably small amount of money on purchasing such a device.

The idea is to provide hardware in the transaction chain which is under the customer's control and is acting on their behalf. This would provide some protection to the customer in the event of a dispute with the bank: an event which is all too common. To avoid increasing the cost or security requirements of the device, the scheme produces transaction logs which can be stored off the device and once created can be verified as complete and authentic.

This is achieved by creating a two-way hash chain within the log and each transaction along with the chain in either direction being authenticated by both the device and the bank. This creates a log chain which is not disputable by either party and which can be verified as internally complete.

In the case of a dispute about transactions more recent than the end of the log, another transaction can be performed in the presence of independent witnesses which will be authenticated by both parties and then 'seal' the log.

In order to create a wider audience for the system the integration with other online financial transactions has also been investigated. There already exist ways to hook the card issuer into the flow of the transaction at the merchant and while current schemes offer very poor security it is a mechanism which could be used to provide a robust solution using the proposed device.

A further barrier to adoption is that while it has been shown that none of the existing systems which were reviewed are completely successful in either their stated goals or in combating all the threats to Internet banking, this may not prevent them being used in practice. Security is a trade-off based on risk. If the banks consider that current defences reduce their risk sufficiently for a lower cost they may think that this is sufficient and not want to deploy a technically superior system.

7.3 Future work

There are few areas in which the research is complete and this is no exception. The proposals made here depend on a number of assumptions as to the feasibility of creating the device and the cost of manufacture. Before any sort of deployment can be considered these are critical questions which need to be answered. From surveying similar technology it is likely to be possible, but that was not the focus of this research.

These proposals have also assumed an attacker model which does not require a lot of physical tamper resistance. This is a trade-off and not ideal, but any increase in tamper resistance requirements will increase the cost. With the advances in polymer electronics there seems to be the possibility of cheap to produce devices with more tamper resistance, but this is still an open question.

More importantly, even with a trusted path there are still possible avenues of attack. The user is often the weakest link in a security chain and this is true here. If the user does not check clearly the information on the device they can still be tricked into authorizing the wrong transactions. The comments in this thesis address some of the issues, but there is still a lot of work to be done in presenting the information in such a way that the user can clearly understand what they are being shown and whether or not it is correct. Some changes to make this easier may need to be internal to the banks and may have other security problems, such as privacy of account holders, problems which will need a lot more thought.

Section 4.6 briefly mentioned the usability of the device proposed in that chapter. Before a device of this sort could be deployed on a wide scale the hypothesis that it is easy to use should be tested. Research into this with a wide selection of users would be valuable.

Chapter 6 investigated other uses to which this research could be put. Integration with Verified by Visa was proposed along with use in more general digital signature schemes. Both these areas could benefit from the results of this research but are beyond the scope of this thesis for more than a superficial mention. More work could be done in both these areas.

Bibliography

[1] Matthew Johnson and Simon Moore. A new approach to e-banking. In Úlfar Erlingsson and Andrei Sabelfeld, editors, *Proc. 12th Nordic Workshop on Secure IT Systems (NORDSEC 2007)*, pages 127–138. University of Reykjavik, Oct 2007. http://www.matthew.ath.cx/publications/2007-Johnson-ebanking.pdf.

[2] Matthew Johnson and Frank Stajano. Implementing a multi-hat PDA. In *The Thirteenth International Workshop on Security Protocols*, number LNCS 4631, pages 295–307. Springer-Verlag, April 2005. http://www.matthew.ath.cx/publications/2005-JohnsonSta-hats.pdf.

[3] Matthew Johnson and Frank Stajano. Usability of security management: Defining the permissions of guests. In *The Fourteenth International Workshop on Security Protocols*, number LNCS. Springer-Verlag, April 2006. To Appear. http://www.matthew.ath.cx/publications/2006-JohnsonSta-Guests.pdf.

[4] Matthew Johnson and Ralph Owen. A real world application of secure multi-party computations. In *The Sixteenth International Workshop on Security Protocols*, number LNCS. Springer-Verlag, April 2008. To Appear. http://www.matthew.ath.cx/publications/2008-JohnsonOwen-Duplimate.pdf.

[5] ISO/IEC. Common criteria for information technology security evaluation. Technical Report 15408, International Organization for Standardization, September 2006. http://www.commoncriteriaportal.org/thecc.html.

[6] Bruce Schneier. Attack trees: Modeling security threats. *Dr. Dobb's Journal*, 24(12):21–29, Dec 1999. http://www.schneier.com/paper-attacktrees-ddj-ft.html.

[7] David N. Card. Learning from our mistakes with defect causal analysis. *IEEE Softw.*, 15(1):56–63, 1998.

[8] Markus Jakobsson. Modeling and preventing phishing attacks. In *Financial Cryptography and Data Security*, volume LNCS of *3570/2005*, page 89. Springer, 2005. http://www.informatics.indiana.edu/markus/papers/phishing_jakobsson.pdf.

[9] Markus Jakobsson and Steven Myers, editors. *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. John Wiley and Sons, 2007. http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0471782459.html.

[10] Gunter Ollmann. The phishing guide: Understanding & preventing phishing attacks, Sept 2004. http://www.ngssoftware.com/papers/NISR-WP-Phishing.pdf.

[11] Evgeniy Gabrilovich and Alex Gontmakher. The homograph attack. *Communications of the ACM*, 45(2), Feb 2002. http://www.cs.technion.ac.il/~gabr/papers/homograph_full.pdf.

[12] A. Costello. Punycode: A bootstring encoding of unicode for internationalized domain names in applications (IDNA). RFC 3492, IETF, March 2003. http://www.ietf.org/rfc/rfc3492.txt.

[13] Eric Johanson. The state of homograph attacks. Technical report, The Shmoo Group, 2005. http://www.shmoo.com/idn/homograph.txt.

[14] Whats in a name: The state of typo-squatting 2007. Report, McAfee, 2007. http://us.mcafee.com/root/identitytheft.asp?id=safe_typo.

[15] Obvious Obscurity. Is it phishing or outsourcing? http://www.obviousobscurity.org/?p=28.

[16] Mikko Hyppnen. 3D spam. *F-Secure Weblog*, Sept 2007. http://www.f-secure.com/weblog/archives/archive-092007.html#00001267.

[17] gadi. Firefox phishing protection bypass vulnerability. Securiteam, Jun 2006. http://blogs.securiteam.com/index.php/archives/467.

[18] Gunter Ollmann. The pharming guide: Understanding & preventing DNS-related attacks by phishers, Aug 2005. http://www.ngssoftware.com/papers/ThePharmingGuide.pdf.

[19] PayPal Help Center. How do I confirm a U.S. bank account already added to my PayPal account? https://www.paypal.com/helpcenter/main.jsp;?t=solutionTab&solutionId=12389&cmd=_help.

[20] Barbara Mikkelson and David Mikkelson. ATM camera. Technical report, Snopes, 2004. http://www.snopes.com/fraud/atm/atmcamera.asp.

[21] Chris Richard. Guard your card: ATM fraud grows more sophisticated. *Christian Science Monitor*, July 2003. http://www.csmonitor.com/2003/0721/p15s01-wmcn.html.

[22] Ali Hussain. Don't use cards at petrol stations. *The Sunday Times*, Feb 2007. http://business.timesonline.co.uk/tol/business/money/consumer_affairs/article1400176.ece.

[23] KeyGhost. KeyGhost SX. http://www.keyghost.com/keylogger.htm.

[24] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320. USENIX, Aug 2004. http://www.usenix.org/publications/library/proceedings/sec04/tech/dingledine.html.

[25] Team Furry. TOR exit-node doing MITM attacks. *MW-Blog*, 2007. http://www.teamfurry.com/wordpress/2007/11/20/tor-exit-node-doing-mitm-attacks/.

[26] Duane Wessels. DNS cache poisoners. lazy, stupid, or evil? In *36th Meeting of the North American Network Operators' Group (NANOG36)*, February 2006. http://www.nanog.org/mtg-0602/pdf/wessels.pdf.

[27] U. Steinho, A. Wiesmaier, and R. Araújo. The state of the art in DNS spoofing. In *4th International Conference on Applied Cryptography and Network Security*, June 2006. http://www.cdc.informatik.tu-darmstadt.de/~rsa/papers/DNS-spoofing-ACNS2006.pdf.

[28] Amit Klein. BIND 9 DNS cache poisoning. Technical report, Trusteer, June 2007. http://www.trusteer.com/docs/BIND_9_DNS_Cache_Poisoning.pdf.

[29] Alla Bezroutchko. Predictable DNS transaction IDs in Microsoft DNS server. Technical report, ScanIT, Nov 2007. http://www.scanit.be/advisory-2007-11-14.html.

[30] Sid Stamm, Zulkar Ramzan, and Markus Jakobsson. Drive-by pharming. Computer Science Technical Report 641, Indiana University, Dec 2006. http://www.symantec.com/avcenter/reference/Driveby_Pharming.pdf.

[31] Various. Router hacking challenge. *GNUCitizen*, Feb 2008. http://www.gnucitizen.org/projects/router-hacking-challenge/.

[32] Jeremiah Grossman and T.C. Niedzialkowski. Hacking intranet websites from the outside. In *Black Hat USA 2006*, Aug 2006. http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Grossman.pdf.

[33] Jeremiah Grossman. Hacking intranet websites from the outside (take 2). In *Black Hat USA 2007*, Jul 2007. https://www.blackhat.com/presentations/bh-usa-07/Grossman/Whitepaper/bh-usa-07-grossman-WP.pdf.

[34] Drew Dean, Edward W. Felten, and Dan S. Wallach. Java security: From HotJava to Netscape and beyond. In IEEE, editor, *IEEE Symposium on Security and Privacy*, pages 190–200. IEEE Computer Society Press, May 1996. http://www.cs.princeton.edu/sip/pub/oakland-paper-96.pdf.

[35] Collin Jackson, Adam Barth, Andrew Bortz, Weidong Shao, and Dan Boneh. Protecting browsers from DNS rebinding attacks. In *14th ACM Conference on Computer and Communications Security*. ACM, Oct 2007. To Appear. http://crypto.stanford.edu/dns/dns-rebinding.pdf.

[36] Dan Kaminsky. DNS rebinding and more packet tricks. In *24th Chaos Communication Congress*, Dec 2007. http://events.ccc.de/congress/2007/Fahrplan/track/Hacking/2393.en.html.

[37] Jochen Topf. HTML form protocol attack. BugTraq posting, Aug 2001. http://www.remote.org/jochen/sec/hfpa/hfpa.pdf.

[38] Obscure. Extended HTML form attack. Technical report, EyeonSecurity, 2002. http://www.hackerz.ir/e-books/Extended%20HTML%20Form%20Attack.pdf.

[39] Adrian Pastor. BT home flub: Pwnin the BT home hub (5) - exploiting IGDs remotely via UPnP. *GNUCitizen*, Jan 2008. http://www.gnucitizen.org/blog/bt-home-flub-pwnin-the-bt-home-hub-5/.

[40] Adrian Pastor and Petko D. Petkov. Hacking the interwebs. *GNUCitizen*, Jon 2008. http://www.gnucitizen.org/blog/hacking-the-interwebs/.

[41] Wikipedia. List of computer worms. http://en.wikipedia.org/wiki/List_of_computer_worms.

[42] Wikipedia. List of computer viruses.
http://en.wikipedia.org/wiki/List_of_computer_viruses.

[43] Wikipedia. List of Trojan horses.
http://en.wikipedia.org/wiki/List_of_trojan_horses.

[44] T. Dierks and C. Allen. The TLS protocol. RFC 2246, IETF, Jan 1999.
http://www.ietf.org/rfc/rfc2246.txt.

[45] Eric Rescorla. Diffie-Hellman key agreement method. RFC 2631, IETF, Jun 1999.
http://www.ietf.org/rfc/rfc2631.txt.

[46] Peter Gutmann. Phishing tips and techniques: Tackle, rigging, and how and when to phish.
http://www.cs.auckland.ac.nz/~pgut001/pubs/phishing.pdf.

[47] Kevin Poulsen. Microsoft vexed by falsified certs. *The Register*, Mar 2001.
http://www.theregister.co.uk/2001/03/23/microsoft_vexed_by_falsified_certs/.

[48] Ken Horn. Apparent phishing by my own bank. http://kendes.blogspot.com/2005/11/apparent-phishing-by-my-own-bank.html, Nov 2005.

[49] Justin Mason. Filtering spam with spamassassin. In *HEANet Annual Conference*, 2002.
http://spamassassin.apache.org/presentations/HEANet_2002/.

[50] Jonathan A. Zdziarski. Concept identification using chained tokens. In *MIT Spam Conference*, 2004. http://www.zdziarski.com/papers/chained.html.

[51] Richard Clayton. Phishing and the gaining of "clue". *Light Blue Touchpaper*, Aug 2007. http://www.lightbluetouchpaper.org/2007/08/16/phishing-and-the-gaining-of-clue/.

[52] Robert McMillan. 'Rock Phish' blamed for surge in phishing. *InfoWorld*, Dec 2006.
http://www.infoworld.com/article/06/12/12/HNrockphish_1.html.

[53] Markus Jakobsson. Distributed phishing attacks. Cryptology ePrint Archive, Report 2005/091, 2004. http://eprint.iacr.org/2005/091.pdf.

[54] Microsoft phishing filter: A new approach to building trust in e-commerce content. White paper, Microsoft Corp, 2005.
http://www.microsoft.com/downloads/details.aspx?FamilyId=B4022C66-99BC-4A30-9ECC-8BDEFCF0501D.

[55] Mozilla. Firefox 2 phishing protection.
http://www.mozilla.com/en-US/firefox/phishing-protection/.

[56] Google. Safe browsing API.
http://code.google.com/apis/safebrowsing/.

[57] R. Rivest. The MD5 message-digest algorithm. RFC 1321, IETF, April 1992.
http://www.ietf.org/rfc/rfc1321.txt.

[58] Kanedaaa. Firefox phishing protection bypass vulnerability. Securiteam, Feb 2007. http://www.securiteam.com/securitynews/5MP0320KKK.html.

[59] Joel Hruska. New URL highlighting feature in Firefox 3 aims to make phishing harder. *Ars Technica*, Jul 2007. http://arstechnica.com/news.ars/post/20070711-new-features-in-firefox-3-aim-to-reduce-phishing-exploits.html .

[60] Opera. Opera security features. http://www.opera.com/products/desktop/security/.

[61] Kanedaaa. Opera 9.10 fraud protection bypass. http://kaneda.bohater.net/security/20061220-opera_9.10_final_bypass_fraud_protection.php.

[62] Tyler Moore and Richard Clayton. Evaluating the wisdom of crowds in assessing phishing websites. In *12th International Conference on Financial Cryptography and Data Security*, Jan 2008. http://www.cl.cam.ac.uk/~twm29/fc08.pdf.

[63] Min Wu, Robert C. Miller, and Simson L. Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM Press, April 2006. http://groups.csail.mit.edu/uid/projects/phishing/chi-security-toolbar.pdf.

[64] eBay. eBay toolbar. http://pages.ebay.com/ebay_toolbar/.

[65] McAfee. McAfee SiteAdvisor, 2007. http://www.siteadvisor.com/.

[66] Richard Clayton. Poor advice from SiteAdvisor. *Light Blue Touchpaper*, Aug 2007. http://www.lightbluetouchpaper.org/2007/08/12/poor-advice-from-siteadvisor/.

[67] Amir Herzberg and Ahmad Gbara. Security and identification indicators for browsers against spoofing and phishing attacks. Cryptology ePrint Archive, Report 2004/155, 2004. http://eprint.iacr.org/2004/155.

[68] Neil Chou, Robert Ledesma, Yuka Teraguchi, Dan Boneh, and John C. Mitchell. Client-side defense against web-based identity theft. In *11th Annual Network and Distributed System Security Symposium*, February 2004. http://crypto.stanford.edu/SpoofGuard/webspoof.pdf.

[69] Waterken Inc. Waterken YURL trust management for humans, 2004. http://www.waterken.com/dev/YURL/Name/.

[70] Ye Zishuang and S. Smith. Trusted paths for browsers. In *Proceedings of the 11th USENIX Security Symposium*. IEEE Computer Society Press, 2002. http://www.usenix.org/events/sec02/full_papers/ye/ye.pdf.

[71] Rachna Dhamija and J.D. Tygar. The battle against phishing: Dynamic security skins. In *Proceedings of the 2005 ACM Symposium on Usable Security and Privacy*, pages 77–88. ACM Press, July 2005. http://people.deas.harvard.edu/~rachna/papers/securityskins.pdf.

[72] T. Wu. The SRP authentication and key exchange system. RFC 2945, IETF, Sep 2000. http://www.ietf.org/rfc/rfc2945.txt.

[73] Blake Ross, Collin Jackson, Nick Miyake, Dan Boneh, and John C Mitchell. Stronger password authentication using browser extensions. In *Proceedings of the 14th USENIX Security Symposium*, page 1732. USENIX, 2005. http://crypto.stanford.edu/PwdHash/pwdhash.pdf.

[74] Two-factor authentication: An essential guide in the fight against internet fraud. Technical Report GP_WP_2W, GPayments, Feb 2006. http://www.gpayments.com/pdfs/WHITEPAPER_2FA-Fighting_Internet_Fraud.pdf.

[75] L. Lamport. Password authentication with insecure communication. In *Communications of the ACM 24.11*, pages 770–772, November 1981.

[76] Daniel L. McDonald and Randall J. Atkinson. One time passwords in everything (OPIE): Experiences with building and using stronger authentication. In *Proceedings of the 5th USENIX Security Symposium*. USENIX, 1995. http://chacs.nrl.navy.mil/publications/CHACS/1995/1995mcdonald-USENIX.pdf.

[77] OUT-LAW. Phishing attack targets one-time passwords. *The Register*, Oct 2005. http://www.theregister.co.uk/2005/10/12/outlaw_phishing/.

[78] RSA Security. RSA SecurID products. http://www.rsasecurity.com/node.asp?id=1311.

[79] VeriSign. Unified authentication. http://www.verisign.co.uk/unified-authentication/.

[80] Bram Cohen and Ben Laurie. AES-hash. Proposal, National Institute of Standards and Technology, May 2001. http://csrc.nist.gov/CryptoToolkit/modes/proposedmodes/aes-hash/aeshash.pdf.

[81] R. Rivest. A description of the RC2(r) encryption algorithm. RFC 2268, IETF, March 1998. http://www.ietf.org/rfc/rfc2268.txt.

[82] John Leyden. Phishers rip into two-factor authentication. *The Register*, July 2006. http://www.theregister.co.uk/2006/07/13/2-factor_phishing_attack/.

[83] EMVCo LLC. *EMV 4.1, Book 4—Cardholder, Attendant, and Acquirer Interface Requirements*, June 2004. http://www.emvco.com/specificationterms.asp?id=4939.

[84] MasterCard International. *Chip Authentication Program—Functional Architecture*, Sept 2004. Available upon request from chip_help@mastercard.com.

[85] Gemalto. Press release, April 2007. http://www.gemalto.com/press/archives/2007/04-18-2007-Barclays.pdf.

[86] Ross Anderson. Yet another insecure banking system. *Light Blue Touchpaper*, Oct 2006. http://www.lightbluetouchpaper.org/2006/10/27/yet-another-insecure-banking-system/.

[87] HSBC. OTP FAQ. http://www.hsbc.com.tr/English/RetailBanking/FAQ/OneTimePassword.asp.

[88] Virus List. Worm.SymbOS.Cabir.a. *Virus Encyclopedia*, June 2004. http://www.viruslist.com/en/viruslist.html?id=1689517.

[89] Trend Micro. SYMBOS_COMWAR.C. *Virus Encyclopedia*, Oct 2005. http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=SYMBOS_COMWAR.C.

[90] Virus List. Virus.WinCE.Duts.a. *Virus Encyclopedia*, July 2004. http://www.viruslist.com/en/viruslist.html?id=1874404.

[91] Charlie Miller, Jake Honoroff, and Joshua Mason. Security evaluation of Apple's iPhone. Technical report, Independent Security Evaluators, July 2007. http://www.securityevaluators.com/iphone/exploitingiphone.pdf.

[92] Melanie Peters. South africa: Mtn moves to prevent sim card swap fraud. *allAfrica.com*, Dec 2007. http://allafrica.com/stories/200801020087.html.

[93] Standard Bank. OTP FAQ. http://www.standardbank.co.za/SBIC/Frontdoor_02_01/0,2354,3447_13906688_0,00.html.

[94] Vassilis Prevelakis and Diomidis Spinellis. The Athens affair. *IEEE Spectrum*, July 2007. http://www.spectrum.ieee.org/jul07/5280.

[95] Risks and threats analysis and security best practices—mobile 2-way messaging systems. Technical report, Mobile Payment Forum, Dec 2002. http://www.mobilepaymentforum.org/documents/Risk_and_Threats_Analysis_and_Security_Best_Practices_Mobile_2_Way_Messaging_December_2002.pdf.

[96] Sarah Hilley. Citibank cuts off bank spies with virtual keyboard. Infosecurity Magazine, Feb 2005. http://www.infosecurity-magazine.com/news/050216_Citibank_keyboard.html.

[97] K.S. Yash. Defeating citibank virtual keyboard protection using screenshot method. *TracingBug*, May 2007. http://www.tracingbug.com/index.php/articles/view/23.html.

[98] Roberto Di Pietro, Gianluigi Me, and Maurizio A. Strangio. A two-factor mobile authentication scheme for secure financial transactions. In *International Conference on Mobile Business*, 2005.

[99] Bryan Parno, Cynthia Kuo, and Adrian Perrig. Phoolproof phishing prevention. In G. Di Crescenzo and A. Rubin, editors, *Financial Cryptography and Data Security*, volume LNCS of *4107*, pages 1–19. Springer-Verlag, 2006. http://sparrow.ece.cmu.edu/~adrian/projects/phishing.pdf.

[100] Cronto. Cronto's visual crytogram. http://www.cronto.com/visual_cryptogram.htm.

[101] Ross Anderson. *Security Engineering*. John Wiley and Sons, Jan 2001. http://www.cl.cam.ac.uk/~rja14/book.html.

[102] PicStop. USB Skype VoIP LCD phone—USB-P10D. http://www.picstop.co.uk/Skype-USB-VOIP-Phone/USB-Skype-Voip-LCD-Phone---USB-P10D.

[103] USRobotics. USRobotics USB internet phone. http://www.usr.com/products/voip/voip-product.asp?sku=USR9600.

[104] Petros Oikonomakos, Jacques Fournier, and Simon Moore. Implementing cryptography on TFT technology for secure display applications. In *The 7th Smart Card Research and Advanced Application IFIP Conference*, volume 3928 of *LNCS*, pages 32–47, April 2006. http://www.cl.cam.ac.uk/~swm11/research/papers/CARDIS2006.pdf.

[105] QRCode. About 2D code.
http://www.denso-wave.com/qrcode/aboutqr-e.html.

[106] Cronto. About Cronto's technology.
http://www.cronto.com/technology.htm.

[107] Specification for the ADVANCED ENCRYPTION STANDARD (AES). FIPS 197, National Institute of Standards and Technology, Nov 2001.
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[108] Morris Dworkin. Recommendation for block cipher modes of operation. Special Publication 800-38A, National Institude of Standards and Technology, December 2001. http://csrc.nist.gov/CryptoToolkit/modes/800-38_Series_Publications/SP800-38A.pdf.

[109] Morris Dworkin. Recommendation for block cipher modes of operation: The CMAC mode for authentication. Special Publication 800-38B, National Institude of Standards and Technology, May 2005.
http://csrc.nist.gov/CryptoToolkit/modes/800-38_Series_Publications/SP800-38B.pdf.

[110] Bruce Schneier. *Practical Cryptography*. John Wiley and Sons, 2003.
http://www.schneier.com/book-practical.html.

[111] British Bankers' Association, APACS, and The Building Societies Association. The banking code, March 2005.
http://www.bankingcode.org.uk/pdfdocs/BANKING%20CODE.pdf.

[112] Philip de Bruin. Creditcard-holders 'must pay'. *News24*, Aug 2004. http://www.news24.com/News24/AnanziArticle/0,,1518-24_1570067,00.html.

[113] Maureen Marud. Phantom withdrawals leave pensioner broke. Technical report, Ombudsman for Banking Services, 2002.
http://www.obssa.co.za/news/17_05_02_2.htm.

[114] Ross Anderson, Mike Bond, and Steven J. Murdoch. Chip and spin.
http://www.chipandspin.co.uk/spin.pdf.

[115] Liz Dolan. Money surgery: Barclays plays fast and loose with chip and pin. *The Telegraph*, Dec 2005. http://www.telegraph.co.uk/money/main.jhtml?xml=/money/2005/12/14/cmliz14.xml.

[116] Ross Anderson. Financial ombudsman losing it? *Light Blue Touchpaper*, Jan 2008.
http://www.lightbluetouchpaper.org/2008/01/23/financial-ombudsman-losing-it/.

[117] Ross Anderson and Nicholas Bohm. FIPR submission to the Hunt review of the financial ombudsman service. Technical report, FIPR, Jan 2008.
http://www.fipr.org/080116huntreview.pdf.

[118] Ross Anderson and Mike Bond. The man-in-the-middle defence. In *The Fourteenth International Workshop on Security Protocols*, April 2006. http://www.cl.cam.ac.uk/~mkb23/research/Man-in-the-Middle-Defence.pdf.

[119] Saar Drimer and Steven J. Murdoch. Distance bounding against smartcard relay attacks. In *Proceedings of the 16th USENIX Security Symposium*. USENIX, August 2007.
http://www.cl.cam.ac.uk/~sjm217/papers/usenix07bounding.pdf.

[120] Bruce Schneier and John Kelsey. Cryptographic support for secure logs on untrusted machines. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium, 1998*, pages 4–4, Berkeley, CA, USA, 1998. USENIX Association. http://www.schneier.com/paper-secure-logs.pdf.

[121] Specification for the SECURE HASH STANDARD. FIPS 180-2, National Institute of Standards and Technology, Aug 2002. http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf.

[122] National Institute of Standards and Technology. Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. *Federal Register*, 72(212):62212–62220, Nov 2007. http://csrc.nist.gov/groups/ST/hash/documents/FR_Notice_Nov07.pdf.

[123] Debra L. Cook, John Ioannidis, Angelos D. Keromytis, and Jake Luck. CryptoGraphics: Secret key cryptography using graphics cards. In *RSA Conference, Cryptographers' track*, pages 334–350, Feb. 2005.
https://db.usenix.org/events/sec04/tech/wips/wips/01-cook-cryptographics.pdf.

[124] Debra L. Cook, Ricardo Baratto, and Angelos D. Keromytis. Remotely keyed CryptoGraphics: secure remote display acces using (mostly) untrusted hardware. In *Proceedings of the 7th international conference on information and communications security*, Dec. 2005.
http://www.ncl.cs.columbia.edu/publications/cucs-050-04.pdf.

[125] International Organization for Standardization. *ISO 7064, Data processing—Check character systems*, 1983.
http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=31531\&ICS1=35\&ICS2=40\&ICS3=.

[126] Victor Boyko, Philip MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using Diffie-Hellman. *Lecture Notes in Computer Science*, 1807:156–172, 2000. http://www.iacr.org/archive/eurocrypt2000/1807/18070157-new.pdf.

[127] Thomas Wu. The secure remote password protocol. In *1998 Internet Society Network and Distributed System Security Symposium*, 1998.
ftp://srp.stanford.edu/pub/srp/srp.ps.

[128] Visa. Verified by Visa. http://www.visaeurope.com/merchant/handlingvisapayments/cardnotpresent/verifiedbyvisa.jsp.

[129] MasterCard. MasterCard SecureCode.
http://www.mastercard.com/securecode/.

[130] Simone van der Hof. Digital signature law survey.
https://dsls.rechten.uvt.nl/.

Appendix A

Implementation details of protocol messages

When presenting the protocols in this thesis an abstract protocol notation was used. This allows reasoning about the protocol independent of any implementation details and makes it simpler to transfer the protocol between different cryptographic primitives as needed.

To actually implement a protocol, however, a more concrete representation is required. This appendix gives all the protocols in the main body of the thesis with field sizes and a concrete implementation.

To be compatible with the device which was used as an example, these figures are all implemented using triple-DES as the block cipher (in CBC mode) and as the MAC cipher in CBC-MAC mode. The hash function is SHA-1.

A.1 Banking dongle protocol

This section contains the protocol from Section 4.5. Each message is given as the format of the block to be encrypted, including padding, followed by the message including the encrypted block. The key used for each encryption is not given here, this can be found in the original protocol notation.

M_1

$0-3$	4	$5 \rightarrow 8$	$9 \rightarrow 12$	$13 \rightarrow 36$	$37 \rightarrow 60$	$61-63$
ID	*0x01*	48	D	K_{BD1}	K_{BD2}	Padding

(1) Key Initialisation packet

0	$1 \rightarrow 4$	$5 \rightarrow 12$	$13 \rightarrow 76$	$77 \rightarrow 84$
0x01	85	IV	Encrypt(M_1)	MAC(M_1)

M_2

$0 \rightarrow 3$	4	$5 \rightarrow 8$	$9 \rightarrow 12$	$13-15$
ID+1	*0x02*	0	D	Padding

(2) Key Initialisation Ack Packet

0	$1 \rightarrow 4$	$5 \rightarrow 12$	$13 \rightarrow 28$	$29 \rightarrow 36$
0x02	37	IV	Encrypt(M_2)	MAC(M_2)

M_3

$0-3$	4	$5 \rightarrow 8$	$9 \rightarrow 12$	$13 \rightarrow L+8$	$L+9 \rightarrow 12$	$\ldots$
N	*0x03*	Len	D	transaction	Type	Padding

Len = sizeof(transaction)+4

(3) Transaction Request Packet

0	$1 \rightarrow 4$	$5 \rightarrow 12$	$13 \rightarrow L-9$	$L-8 \rightarrow 1$
0x03	Len	IV	Encrypt(M_3)	MAC(M_3)

Len = sizeof(transaction) + 38 + sizeof(padding)

Messages (4) and (5) are between the secure the device and the user and are, therefore, not presented here.

M_6

$0 \rightarrow 3$	4	$5 \rightarrow 8$	$9 \rightarrow 12$	$13 \rightarrow L+7$	$L+8 \rightarrow 11$	$L+12$	$\ldots$
N+1	*0x04*	Len	D	Transaction	Type	Auth	Pad…

Len = sizeof(transaction)+5

(6) Transaction Response Packet

0	$1 \rightarrow 4$	$5 \rightarrow 12$	$13 \rightarrow L-9$	$L-8 \rightarrow 1$
0x04	Len	IV	Encrypt(M_6)	MAC(M_6)

Len = sizeof(transaction) + 39 + sizeof(padding)

A.2 Audit protocol

This section contains the modifications to the protocol to produce an audit log. As with the abstract notation in Section 5.3.2 since this protocol shares the first five messages with the previous ones they have been omitted for brevity.

Message (6) is replaced with the message below and then followed by the two additional messages in this protocol.

M_6

$0-3$	4	$5 \rightarrow 8$	$9 \rightarrow 12$	$13 \rightarrow 32$	$33 \rightarrow 52$
N+1	*0x04*	Len	D	$h(O_{I+1})$	$h(L_{I-1})$
$53 \rightarrow L-1$	$L \rightarrow L+3$	$L+4$	$L+5 \rightarrow 12$	$\ldots$	
Transaction	Type	Auth	MAC(transaction)	Padding	

Len = sizeof(transaction)+53

(6) Transaction Response Packet

0	$1 \rightarrow 4$	$5 \rightarrow 12$	$13 \rightarrow L-9$	$L-8 \rightarrow 1$
0x04	Len	IV	Encrypt(M_6)	MAC(M_6)

Len = sizeof(transaction) + 87 + sizeof(padding)

M_6

$0-3$	4	$5 \rightarrow 8$	$9 \rightarrow 12$	$13 \rightarrow 20$	$21-23$
N+1	*0x05*	8	D	MAC(whole transaction)	Padding

(7) Transaction Acknowledgement Packet

0	$1 \rightarrow 4$	$5 \rightarrow 12$	$13 \rightarrow 36$	$37-44$
0x05	24	IV	Encrypt(M_7)	MAC(M_7)

The final message in the protocol is sent to the log.

L_I

$0-3$	$4-7$	$8-11$	$12 \rightarrow 15$
N	I	Len	D
$16 \rightarrow 19$	$20 \rightarrow L-38$	$L-37 \rightarrow 30$	$L-29 \rightarrow 10$
Type	Transaction	M_7	$h(O_{I+1})$
$L-9 \rightarrow 10$	$L+11 \rightarrow 14$	$L+15$	
$h(L_{I-1})$	O_I	Auth	

Len = sizeof(transaction)+57

(8) Log Entry

$0 \rightarrow L$	$L+1 \rightarrow 8$
L_I	MAC(L_I)

Len = sizeof(transaction) + 73

A.3 Unidirectional protocol

Chapter 6 contained variations to the banking dongle protocol. In particular a protocol only depending on secure communications in one direction. The messages sent on the secure channel are given here in detail corresponding to the abstract protocol given in Section 6.2.

As with the audit protocol, the first few messages are the same as in the original protocol. The final message which is used as input to the check digit function is given here:

M_4

$0-3$	4	$5 \rightarrow 8$	$9 \rightarrow 12$	$13 \rightarrow L+8$	$L+9 \rightarrow 12$	$\ldots$
N	*0x04*	Len	D	transaction	Type	Padding

Len = sizeof(transaction)+4

"APPENDIX B"

1. Online banking Java applet.

The Java applet is used to relay communications between the banking server and the USB connected device. Use of a signed Java applet has several benefits. Firstly it does not require special software to be installed on the client computer. The only requirement is a web browser and JVM. These are likely to be installed on any computer which could currently be used for internet banking. Secondly, since USB communication requires a signed Java applet it provides an extra barrier to entry for attackers performing phishing attacks. An unsigned web site is possible to ignore with current web browsers, but signing is required for the Java applet to work.

The Java applet has two modules. The first module connects to the bank and parses the HTTP requests and responses. The second module monitors the USB ports and reads from the pipes connected to the device.

In the demo system there is also a third module. This implements a fake USB backend which talks to the simulated display over TCP. This is also a restricted operation which requires signing, so will behave in a similar fashion to USB communication with respect to signing.

The two modules communicate via concurrency-assured shared queues, one for messages from the web-server to the device and one for replies. The FakeUSB module exposes the standard Java USB interface which the USB Monitor uses to send and recieve data. The structure of the code is given in Figure 1. Dashed messages and objects indicate remote communications.

2. USB monitoring.

This module monitors the pipes to a USB device for transaction reply messages and passes them to the HTTP module for sending to the bank. Messages from the HTTP monitor are written to the USB writing pipe to send them to the device.

```
⟨ ../uk/ac/cam/cl/mjj29/secdis1/applet/USBMonitor.jpp 2 ⟩ ≡
  ⟨ Imports 16 ⟩
  public class USBMonitor extends Thread implements UsbPipeListener
      {
        private Queue⟨byte[]⟩ upqueue, downqueue;      /* Local references to the queues. */
        private Applet applet;      /* Reference to applet to set status messages. */
        public USBMonitor(Queue⟨byte[]⟩ upqueue, Queue⟨byte[]⟩ downqueue, Applet applet)
          /* Create the monitor and keep references to the external objects in use. */
        {
          this.upqueue = upqueue;
          this.downqueue = downqueue;
          this.applet = applet;
        }
        ⟨ PipeListener methods 4 ⟩;      /* Define the methods used to read from pipes. */
        public void run()      /* This method is called in the new Thread */
        {
          try {
            UsbServices services = new FakeUSB();
              /* entrypoint into the USB world, using our fake subsystem. */
            UsbDevice displaydev;
            ⟨ Find Device 3 ⟩      /* Search for the display device. */
            UsbInterface displayiface;
            displayiface = displaydev.getActiveUsbConfiguration().getUsbInterface((byte) 1);
              /* The device communicates on the first interface which has two endpoints. Address 1 is
                 for reading, address 2 is for writing. */
            displayiface.claim();      /* claim the interface. */
            UsbPipe read = displayiface.getUsbEndpoint((byte) 1).getUsbPipe();
            UsbPipe write = displayiface.getUsbEndpoint((byte) 2).getUsbPipe();
              /* Get a handle to the read and write pipes. */
            read.addUsbPipeListener(this);      /* Add a handler to receive read events. */
            read.open();      /* Open the read pipe. */
            ⟨ Watch for downstream events 5 ⟩
              /* loop wating for events to be written and writing them. */
          }
          catch(Exception e)
          {      /* There was some sort of error while communicating with the device. Tell the user that
                    their transactions will no longer work. */
            Debug.print(e);
            applet.setStatus("Error in communicating with the device.", Color.RED);
          }
        }
      }
```

3. Find the secure display device.

The UsbService provider needs to be searched to find our device (rather than any other UsbDevice attached). We check the product string for each device attached to the root hub in turn. If the attached device is a hub, we have to include those devices in the list.

If no suitable device is found then we exit this thread and display an error message on the applet.

```
⟨ Find Device 3 ⟩ ≡
  displaydev = null;
  UsbHub root = services.getRootUsbHub( );      /* Start at the root hub. */
  List⟨UsbDevice⟩ devs = (List⟨UsbDevice⟩)root.getAttachedUsbDevices( ); for ( UsbDevice dev: devs )
    /* Loop over all attached devices. */
  if (dev.isUsbHub( )) {      /* Add devices attached to hubs to the list */
    devs.addAll((List⟨UsbDevice⟩)((UsbHub)dev).getAttachedUsbDevices( ));
  }
  else if (dev.getProductString( ).equals(Applet.DEVICESTRING)) {
      /* If it's a SecureDisplay, get a reference to it. */
    displaydev = dev;
  }
  if (null ≡ displaydev)
  {     /* None found, display an error. */
    applet.setStatus("No␣SecureDisplay␣found,␣cannot␣make␣transactions", Color.RED);
    return;
  }
```

This code is used in section 2.

4. Reading USB events.

Reading data from the USB is done using a UsbPipeListener which is registered with the UsbPipe used for reading. A separate thread reads data from the USB device (run by the USB subsystem), and then data is passed as an event to the UsbPipeListener.

In the listener we simply need to pass the data to the upstream queue to be picked up by the HTTP thread.

```
⟨ PipeListener methods 4 ⟩ ≡
  public void dataEventOccurred(UsbPipeDataEvente)
    /* A transaction response has been read from the device. */
  {
    synchronized (upqueue) {      /* Acquire the upqueue mutex. */
      upqueue.offer(e.getData( ));      /* Add the transaction data to the queue. */
      upqueue.notify( );      /* notify threads blocked on the semaphore. */
    }
  }
  public void errorEventOccurred(UsbPipeErrorEvente)
    /* We are ignoring errors currently, do nothing here. */
  { }
```

This code is used in section 2.

5. Writing USB events.

Writing USB events is simply a matter of passing a data buffer to the UsbPipe used for writing. This buffer is pulled off the downstream queue when one is available.

Since the thread does nothing other than poll the queue, it can block on it, waiting for data to arrive. This is done by polling the queue in a **synchronized** block and if there is none (it returned **null**) then call *wait*( ) on the queue. This will release the mutex and block on the queue's semaphore. When the thread is notified (by another thread calling *notify*( ) on the queue) it will block until it can acquire the mutex again and then go back around the loop to poll again.

The buffer is written to the UsbPipe with a *syncSubmit* call.

```
⟨ Watch for downstream events 5 ⟩ ≡
  byte[]data;      /* Buffer for the events. */
  while (true)      /* Loop forever. */
  { synchronized (downqueue)      /* Aqcuire the downqueue mutex. */
  { data = downqueue.poll( );      /* Poll the queue. */
  if (null ≡ data)
  {
    try {
      downqueue.wait( );      /* Block on downqueue's semaphore. */
    }
    catch(InterruptedException Ie)
    {      /* Ignore this and just poll again */
    }
    continue;
  }
  } Debug.print("Sending␣data");
  Debug.print(data);
  write.open( );      /* Open the write pipe. */
  write.syncSubmit(data);      /* Write the data to the USB device. */
  write.close( );      /* Close the write pipe. */
  }
```

This code is used in section 2.

6. HTTP monitoring.

This module uses an HTTP connection to poll for transaction messages and pass them to the USB subsystem. Messages from the USB subsystem are then passed back using HTTP POST. The messages are sent as application/octet-stream file objects using the literal encoding for the encrypted transaction messages.

```
⟨ ../uk/ac/cam/cl/mjj29/secdis1/applet/HTTPMonitor.jpp 6 ⟩ ≡
  ⟨ Imports 16 ⟩
  public class HTTPMonitor extends Thread
  {
    private Queue⟨byte[]⟩upqueue, downqueue;      /* Local references to the queues. */
    private Applet applet;      /* Reference to applet to set status messages. */
    public HTTPMonitor(Queue⟨byte[]⟩upqueue, Queue⟨byte[]⟩downqueue, Applet applet)
      /* Create the monitor and store references to the queues */
    {
      this.upqueue = upqueue;
      this.downqueue = downqueue;
      this.applet = applet;
    }
    public void run()      /* This method is run in the newly created thread when start() is called. */
    {
      try {
        while (true) {
          ⟨ Poll server 7 ⟩
          {
            applet.setStatus("Receiving␣Transaction␣Request", Color.GREEN);
              /* Show status message during transaction. */
            ⟨ Receive from server 8 ⟩
            applet.setStatus("Transaction␣Request␣Received", Color.GREEN);
          }
          ⟨ Poll upqueue 9 ⟩
          {
            applet.setStatus("Sending␣Transaction␣Reply", Color.GREEN);
            ⟨ Send to server 10 ⟩
            applet.setStatus("Reply␣Sent", Color.GREEN);
          }
        }
      }
      catch(Exceptione)
      {     /* There was some sort of error while communicating with the server. Tell the user that their
            transactions will no longer work. */
        Debug.print(e);
        applet.setStatus("Error␣in␣communicating␣with␣the␣server.", Color.RED);
      }
    }
  }
```

7. Check whether we have a new transaction to read.

We periodically poll the server by connecting and checking for the HTTP response code. The server will send `200 OK` if there is a pending transaction and reply with a mime-type of `application/octet-stream` containing the data. If there is no pending transaction the server will send a `204 No Content` code and a zero-byte reply.

```
⟨ Poll server 7 ⟩ ≡
  URL pollurl = new URL(Applet.POLL_URI);      /* Create a URL class pointing to the download URI. */
  HttpURLConnection pollconn = (HttpURLConnection)pollurl.openConnection( );
    /* Open an HTTP connection to the URL. */
  pollconn.setUseCaches(false);      /* Disable the caches. */
  pollconn.connect( );      /* Open the connection with the new settings. */
  int code = pollconn.getResponseCode( );
    /* The response will be HTTP_OK if there is a pending transaction. */
  if (HttpURLConnection.HTTP_OK ≡ code ∧ 0 < pollconn.getContentLength( ))
    /* Recieve data if there is any. */
```

This code is used in section 6.

8. Read the transaction data from the HTTP reply.

If there is a transaction pending then the transaction length will be encoded in the Content-Length header and supplied as the HTTP content. The data can be read from the connection's InputStream and then appended to the *downqueue* for the USBMonitor to send to the device.

```
⟨ Receive from server 8 ⟩ ≡
  BufferedInputStream is = new BufferedInputStream(pollconn.getInputStream( ));
    /* Get an input stream for the reply content. */
  byte[ ]buf = new byte[pollconn.getContentLength( )];      /* Create a buffer of the correct size. */
  is.read(buf);      /* Fill the buffer. */
  is.close( );      /* Close the input stream. */
  Debug.print("Got␣a␣transaction␣from␣HTTP");
  Debug.print(buf);
  synchronized (downqueue) {
    downqueue.offer(buf);      /* Add the new transaction to the downqueue for sending over USB. */
    downqueue.notify( );      /* Notify threads waiting on this semaphore. */
    Debug.print("Notified");
  }
```

This code is used in section 6.

9. Poll queue for transaction replies.

Polls the *upqueue* for replies which have been recieved on the USB interface. If one has been recieved it sends it to the server.

```
⟨ Poll upqueue 9 ⟩ ≡
  byte[ ]buf;
  synchronized (upqueue) {
    buf = upqueue.poll( );      /* Check the queue for data. poll( ) will remove the next item from the
        queue and return it; or return null if the queue is empty. */
  }
  if (null ≠ buf)      /* Send data if an item was returned. */
```

This code is used in section 6.

10. Upload transaction reply.
If we have any data from the *upqueue* in *buf* send it on to the server.

```
⟨ Send to server 10 ⟩ ≡
  DataOutputStream dos;      /* The output stream used to upload the transaction data. */
  ⟨ Create HTTP Connection 11 ⟩      /* Setup the HTTP connection and get it ready to write to. */
  dos.write(buf);      /* Write the data. */
  dos.writeBytes("\r\n--******--\r\n");      /* Neccessary to finish the HTTP request. */
  dos.flush( );
  dos.close( );      /* Flush and close the connection. */
```

This code is used in section 6.

11. Create HTTP connection.
Java provides classes for doing HTTP connections. To upload a file an *HttpURLConnection* must be created from a `URL`, set the request method and the content-type to `multipart/form-data`. The file can then be written to the *OutputStream* provided by the connection.

```
⟨ Create HTTP Connection 11 ⟩ ≡
  URL url = new URL(Applet.POST_URI);      /* Create a URL class pointing to the upload URI. */
  HttpURLConnection conn = (HttpURLConnection)url.openConnection( );
    /* Open an HTTP connection to the URL. */
  conn.setDoInput(true);
  conn.setDoOutput(true);
  conn.setUseCaches(false);
  conn.setRequestMethod("POST");
  conn.setRequestProperty("Connection","Keep-Alive");
    /* Set the correct options on the connection. */
  conn.setRequestProperty("Content-Type","multipart/form-data;boundary=*****");
    /* Set the form-data content-type and boundry between parts. */
  dos = new DataOutputStream(conn.getOutputStream( ));
    /* DataOutputStream allows us to write strings as bytes easily. */
  dos.writeBytes("--*****\r\n");      /* Separator to start a new component. */
  dos.writeBytes("Content-Disposition:␣form-data;␣name=\"upload\";" +
      "␣filename=\"transaction.data\"\r\n");      /* header describing the component
      type as a file. */
  dos.writeBytes("\r\n");      /* newline to start data. */
```

This code is used in section 10.

12. The applet itself.

The applet itself is created by the web browser's JVM and extends the Applet class provided by Java. The *init* method is called by the JVM to start the applet. This method starts our monitoring threads and then displays a small status message.

Java's Applet class provides access to the parameters the applet was started with, which is used to get the username and authentication token from the bank's web site. It also provides the URL which the applet was loaded from, which is used to construct the URLs for polling and posting transactions.

```
⟨ ../uk/ac/cam/cl/mjj29/secdis1/applet/Applet.jpp 12 ⟩ ≡
  ⟨ Imports 16 ⟩
  public class Applet extends java . applet.Applet
  {
    private USBMonitor um;
    private HTTPMonitor hm;      /* References to the monitoring threads. */
    ⟨ Constants 15 ⟩
    ⟨ Connecting Queues 13 ⟩
    ⟨ Status functions 14 ⟩
    public void init( )
      /* Called from the Applet initialization code. Runs all the init functions and starts the thread. */
    {
      Debug.setThrowableTraces(true);      /* enable stack traces on errors */
      Debug.setLineNos(true);      /* enable line nos on errors */
      Debug.setHexDump(true);      /* enable hex dumps */
      Debug.print("Applet␣Starting");      /* Startup message. */
      setStatus("Initializing...", Color.BLUE);      /* Show a status message */
      URL base = getDocumentBase( );
      POLL_URI = base.getProtocol( ) + "://" + base.getHost( ) + ":" + base.getPort( ) +
          "/bank.cgi?poll=true&user=" + getParameter("user") + "&auth=" + getParameter("auth");
        /* Construct URL to poll for transactions */
      POST_URI = POLL_URI.replaceAll("poll", "post");
        /* Construct URL to post transaction replies to */
      moduleinit( );      /* Create all the modules and queues */
      hm.start( );
      um.start( );      /* Start the threads. */
      setStatus("Waiting...", Color.BLUE);
    }
  }
```

13. Connecting the pieces.

The applet has two main components which monitor the HTTP and USB connections respectively. When they recieve a transaction it must be passed to the other component. This is done by monitoring two queues which are guarded from concurrent access with mutexes.

The mutexes used are the built-in Java mutexes which are guarded using the **synchronized** keyword. Code which accesses either queue will use the construct:

synchronized (*queue*) { *buf* = *queue.poll*( ); } **if** (**null** ≠ *buf*) { *process*(*buf*); }

Each monitoring thread will poll their respective connection and the relevant queue for new traffic. Incoming traffic will be added to the inbound queue; a transaction from the outbound queue will be removed and sent on the connection.

Adding to the queue is guarded in a similar fashion to polling the queue: **synchronized** (*queue*) { *queue.offer*(*buf*); *queue.notify*( ); }. Notice that there is also a call to *notify*( ), which unblocks any threads waiting on the semaphore on that object.

⟨ Connecting Queues 13 ⟩ ≡

```
private Queue⟨byte[]⟩upqueue;      /* Reference to the upstream queue. */
private Queue⟨byte[]⟩downqueue;    /* Reference to the downstream queue. */
public void moduleinit( )      /* Initializes the modules, queues and connects them together. */
{
  upqueue = new LinkedList⟨byte[]⟩( );
  downqueue = new LinkedList⟨byte[]⟩( );      /* Create the queues. Transactions do not need to be
      understood by this module, just passed along as raw byte arrays. */
  hm = new HTTPMonitor(upqueue, downqueue, this);
  um = new USBMonitor(upqueue, downqueue, this);
    /* Create the modules, pass in references to the queues to connect them. */
}
```

This code is used in section 12.

14. Displaying status messages.
The applet has a status line visible to the user which updates them on the progress of the transactions.

```
⟨ Status functions 14 ⟩ ≡
  private String status;      /* Stores the current status. */
  private Color bgcol;        /* Stores the background colour for the current status. */
  public void setStatus(String status, Color c)      /* Changes the status message. */
  {
    Debug.print("Setting Status: " + status);
    this.status = status;      /* Set the status */
    this.bgcol = c;
    repaint();      /* Force an update of the display */
  }
  public void paint(Graphics g)
    /* Called to update the Applet window. Displays a status message. */
  {
    Debug.print("Repainting window with: " + status);
    super.paint(g);      /* Make sure all sub-components are drawn. */
    g.setFont(new Font("Times New Roman", Font.BOLD, 16));
    g.setColor(bgcol);
    g.fillRect(0, 0, 400, 80);
    g.setColor(Color.BLACK);
    g.drawString(status, 40, 40);      /* Display the status. */
  }
```

This code is used in section 12.

15. Configuration constants.
Various system-specific configuration is required to the applet. This will depend on the device being used and the web site the system is securing. They are set here to the values for our concept demo system.

```
⟨ Constants 15 ⟩ ≡
  public static String POST_URI;      /* URI for sending events to the server. */
  public static String POLL_URI;      /* URI for checking for events from the server. */
  public static String DEVICESTRING = "FakeUSB Device";
    /* Device string to search for to identify the display. */
```

This code is used in section 12.

16. Imports.
All the library classes which are used are imported into the files explicitly.

⟨ Imports 16 ⟩ ≡

```
package uk.ac.cam.cl.mjj29.secdis1.applet;      /* Declare the package these classes are in. */
import cx.ath.matthew.debug.Debug;      /* Debug library. */
import java.awt.Graphics;      /* Used to draw to Applet components. */
import java.awt.Color;      /* Superclass of all colours for drawing to applet components. */
import java.awt.Font;      /* Used to set the font for drawing to applet components. */
import java.io.BufferedInputStream;      /* Buffers downloaded data for reading. */
import java.io.DataOutputStream;      /* Converts various types to bytes when writing to streams. */
import java.net.URL;      /* Creates connections to URLs. */
import java.net.HttpURLConnection;      /* Wraps a socket to parse HTTP requests and replies. */
import java.util.Queue;      /* LIFO interface to a collection for the up and downstream queues. */
import java.util.LinkedList;      /* Pointer-based collection implementation. */
import java.util.List;      /* List interface to collections. */
import javax.usb.UsbServices;      /* The USB point of entry. */
import javax.usb.UsbHub;
import javax.usb.UsbDevice;
import javax.usb.UsbInterface;
import javax.usb.UsbPipe;      /* USB classes for accessing devices. */
import javax.usb.event.UsbPipeListener;
import javax.usb.event.UsbPipeDataEvent;
import javax.usb.event.UsbPipeErrorEvent;      /* USB classes for handling read events. */
```

This code is used in sections 2, 6, and 12.

17. Index.