Toward a Broader View of Security Protocols

Matt Blaze
Department of Computer and Information Science
University of Pennsylvania
blaze@cis.upenn.edu

6 March 2004 – Preproceedings Draft for SPW12 (Cambridge, 2004)

Computer and network security researchers usually focus on the security of computers and networks. Although it might seem as if there is more than enough insecurity here to keep all of us fully occupied for the foreseeable future, this narrow view of our domain may actually be contributing to the very problems that we are trying to solve. We miss important insights from, and opportunities to make contributions to, a larger world that has been grappling with security since long before the computer was invented.

This position paper initiates and advocates the study of "Human-Scale Security Protocols" as a core activity of computing and network security research. The *Human-Scale Security Protocols (HSSP)* project treats "human scale" security problems and protocols as a central part of computer science. Our aim is to identify, stimulate research on, analyze, and improve "non-traditional" protocols that might either have something to teach us or be susceptible to improvement via the techniques and tools of computer security. There are compelling security problems across a wide spectrum of areas that do not outwardly involve computers or electronic communication and yet are remarkably similar in structure to the systems computer scientists routinely study. Interesting and relevant problem spaces that computer security has traditionally ignored range from the very serious (preventing terrorists from subverting aviation security) to the trivial and personal (ensuring that a restaurant serves the same wine that was ordered and charged for).

We use the term "human-scale security" to refer here to high-level security protocols (such as commercial transactions) performed manually by people and to systems and objects intended for direct human interaction (such as mechanical access controls and paper documents). It is distinct from (but related to) the study of the various financial and legal protocols that are analyzed in economic terms, e.g., with a game theoretical model of behavior, and with the aim of designing systems that encourage fair play. Rather, we are concerned here with the often informal protocols that have evolved to prevent outright cheating or crime, as well as with the (often *ad hoc*) non-computerized security mechanisms and practices that protect the physical world. An important characteristic of these protocols and systems is that their design and operation do not depend on, and are not motivated by, electronic computers or communications systems.

Human-scale security systems are relevant to us first because they form the basis (the "root") of trust in the complex systems used for society's basic functions. The trustworthiness of any system (computerized or not) ultimately depends on the integrity and reliability of the people who built and run it, on the security of physical objects, on the soundness of information, and on procedures carried out by human beings (who employ whatever explicit or implicit interfaces the system provides). Yet we often have only informal, *ad hoc* metrics for the security of these basic system elements, and even less of an understanding of how their security properties compose than we do for computing systems. These systems often fail in ways that mimic common security breaches in computers, with similar results and for similar reasons.

Secondly (and conversely), for all their *ad hoc* properties, human-scale security systems appear to have much to teach us. Protocols and systems implemented and used directly by people tend to be heavily optimized for efficiency as well as for performance against the perceived threat model and risk. Their evolutionary development process is often much slower (and more informal) than that used to produce computer systems, with optimizations typically discovered by users seeking to reduce their effort and expense and with countermeasures against specific vulnerabilities introduced only after attacks become a perceived practical risk. Some of the resulting protocols seem to be quite good, perhaps close to optimal for their applications.

The study of human-scale security seems to have much to offer computer and communications security research and practice. How secure are these protocols when analyzed with the methods and against the threat models of computer security? How can well-optimized and highly risk-sensitive human scale systems be adapted to improve computer security protocols? How do human-scale security elements compose? How do they interact with computer security? Can we adapt the trust management techniques from computing to specify and enforce better security in human-scale systems?

1 Human-Scale Security Protocols

There has been relatively little work on human-scale protocols by computer scientists and cryptologists, at least in the half century since we became distracted by the invention of the electronic computer. However, the relatively few example of serious computer science and cryptologic investigation into the subject that do exist are in fact quite encouraging.

Two recent papers from the computer science literature illustrate the kinds of analysis advocated here. The first provides a striking example of how an obvious attack from the physical world can expose a much less obvious, yet fundamentally similar, vulnerability in computer networks. The second introduces an attack against a human-scale system that seems entirely obvious when examined in computer security terms but that remains quite obscure otherwise.

1.1 Denial of Service and Burglar Alarms

Our example from the former category is the delightful CACM paper (and keynote address) by Needham ten years ago on the problem of denial of service in (physical world) burglar alarm systems [Nee94]. The central insight here was that an alarm system can be compromised in two ways: the direct and most obvious approach, which involves preventing the alarm signal from being sent (e.g., by cutting wires), and a more subtle, indirect approach, which involves overwhelming the system's capacity to respond by repeatedly triggering perfectly valid alarm signals.

To perform this second kind of attack, a burglar might set off the alarm (e.g., by rattling the door), wait in the bushes for the police to respond, and set the alarm off again as soon as they leave, repeating until the police stop responding altogether. In other words, Needham observed that we can deny service not only by preventing the system from working, but by letting it do exactly what it was designed to do. (Here the attack seems *more* obvious when the system is analyzed in physical terms than it does when analyzed in abstract terms, since anyone actually installing, responding to, or trying to defeat an alarm system would intuitively consider the capacity of the police to respond as an integral part of the system). The attack illustrates a tradeoff in the design of such systems: greater sensor sensitivity increases security only until it becomes possible for an attacker to deliberately manipulate them, at which point it becomes a serious liability.

The subsequent recent DoS attacks on the Internet exploit a similar property of the IP architecture: one need not cut the wire associated with a network link when one can simply persuade the hosts that are connected to that wire to overwhelm one another with traffic.

1.2 Privilege Escalation in Mechanical Locks

Our example from the later category is drawn from our own initial work on human-scale protocols: the privilege escalation vulnerabilities in master-keyed mechanical locks[Bla03]. Here, the central insight is the modeling of a conventional master-keyed pin-tumbler lock as a online authentication oracle. This is a natural first step from a computer security perspective; a lock is, after all, an "online" service that accepts or rejects keys. An obvious question to ask is whether the "advertised" interface to the service can be exploited to leak information about its secrets (in this case, the configuration of the valid key).

Ordinary (single-key) locks appear to do well under such an analysis. A lock (an ideal, *abstract* lock without mechanical imperfections, at least) will not open unless all of its tumblers are set to their correct secret depth by the key. The "oracle" gives the same response to an "almost correct" key (one that sets most of the tumblers to the correct depth but has at least one at the wrong depth) as it does to a "completely incorrect" key (one that sets all tumblers to the wrong depth). The best attack against an ideal lock appears to be exhaustive search of the keyspace. This should be reassuring for lock users, since this search would be exponential in the number of tumblers. Lock attackers must resort, therefore, to exploiting vulnerabilities in the implementation (e.g., by "picking" the lock).

Master keyed locks, on the other hand, immediately crumble under such an analysis. (A master keyed lock system is one in which locks can be opened by at least two keys, a unique "user" key that opens a single particular lock, and "master keys" that can open all or some of the locks in the system). Here the oracle can be exploited efficiently and easily to solve the (ostensibly exponential) problem of amplifying the rights of an ordinary user key into those of a system-wide master key

The details of the attacks are beyond the scope of this paper, but depend on the fact that the tumblers in master-keyed mechanical locks have two correct (and secret) depths, one for the user key and another for the master key. The tumblers are "evaluated" independently of one another when a key is inserted; the lock will open with a key cut completely as a user key, completely as the master key, or *with some tumbler positions cut for the user key and others cut for the master key.* This latter property allows the holder of a user key to issue a small number of queries to the lock (via specially cut test keys) to learn the master key's tumbler depths one by one.

As obvious as this attack may be in computational terms, it is in fact quite obscure if locks are modeled only in mechanical terms, as they have been for hundreds of years. Most computer scientists and cryptologists quickly see several such attacks once the problem is set up and described in this way, but it requires remarkable and unusual insight for a locksmith to discover or recognize any problem at all, given the mechanically-oriented analytical tools ordinarily employed in that domain. (And indeed, it isn't entirely clear how widely known this attack was within the trade; some locksmiths claim not to have known about it all, while others assert that "everyone" who "needed" to know such things already did. All we can be sure of is that any locksmiths and lock makers who did know of the attack must not have considered it to be very serious, since they failed to warn potential users about it.)

An interesting property of this attack is that it arises from a basic design weakness in how locks are master keyed, not from any flaw in their implementation or manufacture. Any lock master keyed with the standard techniques, no matter how well-made, suffers from this vulnerability; it is a analogous to a "protocol failure" rather than to an "implementation bug."

This small result is encouraging; aside from exposing a practical yet little known weakness in locks, it provides an accessible demonstration of several otherwise rather abstract computer security principles. It suggests that the methods and tools of computer security have something to contribute beyond the security of computers.

2 Some Human-Scale Security Problems

Aside from the examples in the previous section, it is not difficult to find human-scale systems whose characteristics and requirements are similar to those of computing and communications security.

What follows is a rather scatter-shot sampling of human-scale security problems that might be susceptible to a computer science style of analysis or that seem to have something to teach us (or, perhaps in most cases, both). These examples are intended merely as starting points to illustrate the relationship between computer security abstractions (and vulnerabilities) and human scale systems and to stimulate further discussion and research, not as as a comprehensive survey of the problem space.

2.1 Ordering wine in a restaurant

The procedure in restaurants for ordering a fine (or not so fine) bottle of wine follows a largely standardized set ritual over several rounds of interaction. The wine ordering problem appears to have several familiar (and difficult) requirements: secret communication (not revealing to eavesdroppers – the guests at the table – the price range of the wines being advised about), integrity (ensuring that the correct bottle is delivered, opened and not switched for a cheaper product), and non-repudiation (making it difficult for the customer to claim that he or she had really ordered, and believed to have consumed, a cheaper bottle when the bill is presented).

The protocol appears to have evolved over many years, and seems to be well optimized for efficiency (it requires only a few round trips) and against practical threats (to the point that today hardly anyone seriously worries about fraudulent wine downgrades, although it must have been a problem at some point in culinary history). There seems to be much to admire about this protocol, and it appears to be amenable to more formal notation and analysis. What can we learn from it?

2.2 Paying the check in a restaurant

As well-optimized as the restaurant wine ordering protocol may be, the familiar protocol for settling the bill with a charge card is less encouraging. The common protocol requires at least six “flows” between the server and the patron (three round trips):

$P\,(patron) \rightarrow S\,(server)$	:	Request bill	(1)
$S \rightarrow P$	:	Calculate and present bill to P	(2)
$P \rightarrow S$	:	Examine bill; if incorrect complain and have bill recalculated; if correct summon server	(3)
$S \rightarrow P$	:	Collect bill and card from P; run charge; return card, charge slip and bill to P	(4)
$P \rightarrow S$	:	Examine charge slip and bill for consistency; if incorrect complain and have charge invalidated; if correct sign and summon server; exit	(5)
$S \rightarrow P$	:	Collect signed charge slip	(6)

This protocol involves separate interactions for summoning the server to obtain the bill, examining the bill, summing the server to collect the the credit card, examining the charge, and returning the signed charge slip. Intuition suggests that these steps are needed to protect the patron against incorrect charges and the restaurant against invalid or over-limit cards.

In fact, at least one round trip can be eliminated and its steps collapsed without any apparent loss of security for either party:

$P \rightarrow S$	:	Request bill and present card to S	(1)
$S \rightarrow P$	:	Calculate bill and run charge with card; present bill and charge slip to P	(2)
$P \rightarrow S$	:	Examine bill and charge slip; if incorrect complain to have bill recalculated and charge invalidated; if correct sign charge and summon server; exit	(3)
$S \rightarrow P$	:	Collect signed charge slip	(4)

This smaller protocol appears to have essentially the same properties as the conventional protocol. It provides the same opportunities for errors to be detected and corrected, if at the expense of requiring a charge invalidation for any detected errors in the bill. But here the patron need wait for only one round trip and can leave at step (3). This “obvious” optimization is apparently not so obvious to the users of the protocol, given the motivation of parties in both roles to reduce latency and idle time.

2.3 Railroad seat checks

Many American (and perhaps other) passenger railroads employ on-board ticket collectors, who sweep through trains as they pass between stations and exchange the tickets of newly-boarded passengers for *seat checks* that indicate their paid destinations. The ticket collector collects and destroys the seat check just prior to arrival at the stations for which they are encoded. (Passengers without seat checks can therefore be presumed to have just boarded and are asked for their tickets).

The encoding of seat checks is an interesting problem, since they need to be easy to read at a glance,

easy to encode, and must make it difficult to modify the encoded destination to one farther (and more expensive) than the one for which originally issued. A rather ingenious encoding has evolved, passed down entirely as folklore among railroad workers and used extensively in practice on many railroad lines. Seat checks are small printed slips of cardstock paper. An unmodified card indicates the train's final (and most expensive) destination. A small vertical tear halfway down the card's length indicates the second to last stop. A torn off corner indicates the third stop. And so on, with the seat check more and more mutilated for closer (cheaper) destinations. The encoding has the intended property that it is easy to read and encode quickly but difficult to modify to obtain fraudulent travel to a farther station; it implements a paper-based monotonically decreasing counter.

There is a vulnerability in the seat check protocol as described, however. It is possible to obtain unlimited travel to any nonterminal station simply by purchasing a ticket for one station beyond the attacker's intended destination and retaining the (uncollected) seat check for use on the next trip.

Although this could be fixed, e.g., by encoding the date on the seat check, the protocol has another problem familiar to computer security practice: it cannot be easily upgraded. Because it is passed on only as folklore, as opposed to being written into standard procedure, the railroads have no centralized mechanism for deploying a newer, more resilient scheme should seat check fraud ever become an actual problem. Only when the users (train crews) themselves recognize the problem does the protocol have any chance of evolving or being upgraded.

2.4 Smuggling and aviation security

A number of the protocols used to prevent the introduction of contraband (e.g., weapons) on commercial flights have surprisingly subtle vulnerabilities if not analyzed and implemented very carefully. For example, prior to the tighter security imposed after September 11, 2001, it was possible to smuggle a weapon onto a domestic U.S. flight by exploiting a weakness in the *extra* security given to international arrivals and without requiring the failure of any security checkpoint to provide proper screening. The vulnerability was simple: the attacker checked in in, say, London, which has good security screening of passengers and luggage. On arrival in, say, New York, the attacker clears U.S. Customs and connects to a domestic flight. The protocol allowed a weapon to be smuggled onto the final flight, because the Customs screening (an extra security layer) allowed passenger access to checked luggage. Although the passengers may have been screened properly for weapons an London, the luggage was screened only for explosives, and it was possible to remove a weapon from the luggage (in the Customs hall) prior to boarding the domestic connecting flight.

A solution is to require passenger screening after clearing Customs. This appears to have been known to the security authorities, but not uniformly implemented or understood at a local level; post-customs passenger screening was not universally required or enforced at US airports prior to September 2001.

Other complex interactions still exist in the composition of customs inspections and aviation security that may permit various kinds of smuggling. In particular, the handling of lost baggage appears to permit a smuggler to avoid risk under certain circumstances.

The techniques of cryptographic protocol analysis seem directly applicable to aviation security and customs inspection protocols, and a more systematic analysis could possibly detect other, as yet undiscovered, vulnerabilities or expose and make explicit the requirements and assumptions.

2.5 Drug testing in sports

Certain sports players (e.g., in professional and amateur competition) are subject to random or routine screening for performance-enhancing chemicals. The protocols for these tests are surprisingly involved, and appear to have been designed to prevent both cheating by the player (substituting test samples or spoiling the test validity) and the tester (who might attempt to either aid or "frame" a player by substituting a tainted sample). The complexity of these protocols raises serious questions as to their soundness; they seem ripe for analysis and study.

2.6 Voting in democratic elections

There is considerable controversy in the United States surrounding the introduction of computerized voting systems to replace mechanical and paper ballots in national and local elections. The lively debate focuses on whether various proposed products and systems are trustworthy enough, and indeed on whether computerized systems can ever meet the security, privacy and robustness requirements of democratic elections. Many computer science and cryptologic researchers have embraced this problem, contributing both technical analysis as well as political commentary to the debate.

The central question on which both the current scientific research as well as the public policy debate on electronic voting focuses is how computerized systems can be made to mimic the properties of traditional physical ballots. However, this leaves behind a number of equally interesting, and critically important, questions about the underlying application.

How secure are paper ballot systems in the first place? How do they fail? How do they scale? How well do they meet their stated security requirements[Com02]? Many of the recently developed cryptologic techniques and models for analyzing anonymous communication systems, distributed computation, and so forth, would seem to rather directly applicable here.

3 Human-Scale Security in Mainstream Computer Science Research

To the extent that human scale protocols appear in the computer science literature, they are usually in the form of (often strained) analogies illustrating some computational principle rather than as the focal point of evaluation. Several areas of recent computing research, however, touch upon or are relevant to human-scale protocols.

3.1 Trust Management

Trust Management[BFL96][BFS98][BFIK99a], introduced in 1996, is concerned with the specification, checking for compliance, and enforcement of security policy in complex, decentralized systems. Although existing trust management systems, languages and tools operate at a relatively low level of technical policy (e.g., [Ker01]), the framework and mechanism appears to be useful for managing security policies in systems with human-scale components.

3.2 Security Engineering and Software Vulnerabilities

Many aspects of software systems research are concerned with identifying, repairing, and preventing, implementation errors ("bugs") that introduce security vulnerabilities. Much of the motivation for such advances as restricted execution environments, "safe" program languages (e.g., Java), and dynamic and static program memory analysis tools, is the prevention or containment of errors that allow software to be misused in unexpected and dangerous ways. Much of modern software engineering research on program correctness is now focused particularly on the security implications of incorrect programs.

A related, and relatively recent, branch of computing sometimes called *security engineering*, focuses on the development and application of sound engineering practices for building systems with good security properties. Anderson's treatment of the subject, for example [And01], views security in broad terms, drawing on many human-scale systems for inspiration and as examples. Similarly, US DoD security standards often include specific human security requirements (e.g., *two-person control* for access to certain systems).

However, human-scale systems are generally regarded by security engineers as providing accessible examples, rather than as focal points for study or as integral parts of the system. In particular, no general security engineering metrics for evaluating or quantifying the security of human-scale system components are in common use, and those that do exist appear to be drawn at least as much from folklore and intuition as from engineering experience.

3.3 Trusted Computing and Secure Booting

So-called "trusted computing" architectures (such as TCPA, etc), aim to provide a "secure" computing environment based on a small tamper-resistant cryptographic module. Certain smartcard-based applications (e.g., [BFN98]) attempt similar functionality for specific applications (such as electronic cash, file decryption, etc.)

An essential element is of this area is the notion of bootstrapping from small trustworthy elements into complex systems. Secure booting (e.g., [Arb99]) uses this concept for general computing, but the idea can be applied more generally.

Human-scale systems frequently exploit a very similar kind of "trust amplification" in which untrustworthy components are contained or managed by more trustworthy ones.

3.4 Hardware Security and Cryptographic Key Material Management

The security of especially sensitive data (particularly cryptographic key material) is often delegated to "tamper-resistant"hardware. This hardware is often assumed to be secure by software designers, but recent work (e.g., [AK96] [Kuh98] [Kuh02] [KJJ99] [Koc96]) suggests that such assumptions are frequently proven to be unjustified.

An unfortunate side effect of assumptions that secure hardware is indeed secure (because we lack the tools to analyze it) is that when they are discovered to be wrong (whether by the analyst or the attacker), the result is generally catastrophic for the systems that use it. Often the entire security model rests on the assumption that the sensitive data managed by "secure" devices cannot possibly be compromised, and often there is no viable recovery plan should these assumptions prove false.

Similarly, human-scale protocols and systems are often assumed (without justification) to be trustworthy or abstracted out of the security analysis of complex computing systems that include them. A more

integrated security analysis toolkit might accommodate more diverse system elements, or at least allow for the possibility of failure in human-level components.

3.5 Economics and Risk Management

There has recently emerged a vibrant research community focused on the relationship between economics and computer science, often either investigating security problems directly or drawing talent from the traditional computer security and cryptology research communities.

Economics is especially relevant to human-scale security protocols not only because commercial transactions motivate many of the problems, but because the parties in these protocols can often be assumed to follow economic models of behavior. For example, high latency and idle waiting times are usually not well tolerated by people, and are likely to cause perceived un-needed rounds of interaction to be optimized out (intentionally or not) from some protocols, even at the expense of reducing security and increasing risk.

3.6 Human Factors

Almost all security systems have a user interface of some sort, but *Human factors* research on computer security has been remarkably limited. Although such research is beyond the scope of this paper, it acutely highlights the need for it.

4 Future work: The Human-Scale Security Protocols Project

We believe human-scale systems are interesting and important. In this section, we speculate on some future research directions for a "Human-Scale Security Protocols (HSSP)" effort. The objective of this project is to establish the study of human-scale activity, applications, and systems as an integral part of computer security research and practice. In particular, we expect to use HSSP as a starting point to motivate three (complementary) courses of research:

- Analyzing the security properties (especially with regard to how they succeed and fail) of a range of human-scale protocols and applying the lessons learned toward improving computer security methodologies. (What can computer science learn from these systems?)
- Applying the tools and techniques of computer security and cryptology in novel ways to analyze, attack and improve the security of human-scale systems. (What can computer science do for human-scale security systems?)
- Developing stronger ties between the security properties of underlying system components (including their physical environment, hardware, software and networks) and the requirements of the applications that are built upon them ("bootstrapping" trust). (What is the relationship between human applications and the security of the systems that implement them? Can their requirements and properties be quantified or at least made more explicit?)

The dual nature of the project's objective – establishing a sound intellectual foundation for the study of human-scale security in computer science as well as making specific contributions to the security of various systems – means that some research here must be exploratory in nature while others must be focused on specific seminal subproblems.

4.1 Taxonomies of threats and security properties of human-scale applications

It seems especially useful to establish comprehensive "taxonomies" of threats (and risks) in human-scale applications, as well as of the security properties that are required of or that characterize the systems that implement them. Such taxonomies would be useful for analysis of both human- and computer- based systems (and especially systems based on a combination of the two) in a coherent, more uniform manner.

A longer term (and more ambitious) goal is to develop prudent security engineering practices that can integrate both kinds of systems, without requiring separate treatment of between kinds of components. Existing work, e.g., [AN96] or [ASSW02] on security protocol engineering does not seem to be completely applicable to the design of hybrid systems that include human scale components.

4.2 Trust Management at the Human Scale

Our previous work defined the trust management and compliance checking problems as a model for security policy and enforcement and established an abstract "trust management layer" as a system security element in distributed systems.

We are expanding our study of trust management to make it a more useful abstraction for human scale applications.

A basic principle of our trust management tools (such as PolicyMaker [BFL96] and KeyNote [BFIK99b] [BIK03]) has been the *decentralized* nature of the policies for which they check and enforce compliance. That is, a policy can be made up of signed assertions from many different entities, who need not know the details (or even of the existence) of the top level policies of the ultimate endpoints that use them.

This property makes it relatively simple to adapt these systems to new applications, provided that the security policy enforcement points are capable of executing the trust management engine. This is not usually a problem for even very modest computing platforms (e.g., KeyNote runs even on early-generation Palm Pilots), but obviously makes it difficult to integrate into human-scale protocols that depend on people to carry out parts of the task. In particular, the basic service of the trust management system is *compliance checking*; it allows or disallows access, but does not have any means for providing output to or receive input from a human user during the actual evaluation process.

This seems an unfortunate limitation, since many of the most important security critical systems (with complex security policies) include a human-scale component. For example, imagine a comprehensive physical security access policy, which includes both electronic access controls as well as human guards who check identification and interrogate individuals who request access, with different identity requirements for different kinds of access under different circumstances. It would be useful express as part of the security policy the identity documents and questions for the guards to ask under different circumstances, and for the trust management engine to be able to display these questions (and receive the answers) during the compliance checking process. Current tools do not support such a model of interaction; all input to the query is passed to the compliance checker as a "batch," returning a single answer based only on the query.

Indeed, we are aware of no security policy management tools that can capture interactive human-scale operations along with those enforced directly by people.

To make trust management tools useful for such applications, we intend to extend the KeyNote toolkit to allow interactive negation as part of the query evaluation process. In addition, we intend to develop KeyNote trust management policies that capture a variety different human-scale security scenarios. In particular, many human scale protocols have implications not just for security, but for privacy as well. The

security and privacy requirements may be in conflict (as discussed, e.g., in the previous section).

4.3 Analysis of Specific Human-Scale Systems

We are examining several specific human-scale systems, with the aim not only of investigating their security properties, but also to motivate and identify other problems for subsequent, larger-scale study.

4.3.1 Small-scale transaction protocols

Small-scale transactions (such as, e.g., the restaurant rituals discussed in the previous section) are a rich source of human-scale security requirements, protocols, and problems. In addition to the protocols discussed in the previous section, we will identify other appropriate transaction protocols, formalize their apparent requirements, identify the security properties of their components, and analyze the existing protocols as well as alternative versions. Of particular interest for these protocols is a better understanding of how they fail and of their relationship to risk management strategies.

4.3.2 Physical Access Controls

Physical access controls (e.g., locks, safes and vaults [Tob00], [Oeh97]) are a rich source of security requirements and designs. As previously discussed here and in [Bla03], mechanical locks are in fact readily modeled as computers, and indeed, are being replaced or managed by embedded micro-controllers in many applications. Traditional analysis of these systems has focused on their physical properties and implementation (e.g., vulnerability to forced entry or lock picking), rather than on abstract properties of the design or the application.

Inexpensive and low-power embedded micro-controllers have been used in security and access control systems for many years. Electronic locks, of course, do not have mechanical tumblers and are therefore not usually directly vulnerable to mechanical manipulation. Of course, this does not at all imply that electronic locking systems are inherently more secure than their mechanical counterparts. The underlying lock mechanism is still mechanical and may be subject to mechanical bypass. The electronic control mechanism may be vulnerable to attack, e.g., through the introduction of RF or power faults or via emission monitoring. And, of course, electronic locks have at their root software of increasing size and complexity as they become more sophisticated (and as they are networked into centralized control systems). Needless to say, the software used in electronic lock systems is no less subject to bugs, vulnerabilities and protocol failures than the typical (buggy) software of similar complexity used in other applications. Indeed, security system software may well be considerably worse, since it is often purpose-written and may be subject to only limited scrutiny and testing.

We expect to continue our study of master-keyed mechanical locks. An open problem in [Bla03] was how to better structure the keyspace of master keyed locks to limit the information available to the attacker who can either disassemble a lock or use a lock as an oracle. Preliminary results suggest that although there are fundamental tradeoffs that limit the size of systems that can be implemented without leaking information to the attacker, it is possible to build medium-scale master keyed lock systems with “false” cuts that resist our attack and are otherwise secure.

We can also expand this study to include software access control systems, with the aim of identifying software, hardware, and interface interactions that can lead to vulnerabilities similar to those occurring in

general purpose computers and networks. Conversely, we will attempt to identify design strengths in these systems, especially defenses against known attacks, that can be generalized to apply to general computing systems. (The limited interfaces of access control systems is sometimes cited as a strength compared with general purpose computers, but results such as those of Anderson, Kocher, and Kuhn perhaps suggest otherwise).

4.3.3 Sensors and Alarm Systems

As noted in the previous section, alarm systems (and sensor system networks generally) provide an interesting framework and motivation for a number of computer and network security problems, most notably denial of service. The convergence toward the use of general purpose platforms (computer and network) for these applications makes the relationship between computer security and physical security all that much more relevant.

Furthermore, as alarm and area security systems collect and record more and more data, the interactions and tensions between security policies (which encourage collecting and storing more data) and privacy policies (which encourage ignoring or destroying the same data) become more acute. We hope to develop an integrated policy model that can capture security as well as privacy requirements at the same time (e.g., trust management for privacy and security simultaneously).

5 Conclusions

We have have focused too much of our attention on computers and electronic communication systems. Small-scale, mechanical, physical, and paper security systems deserve our attention and scrutiny.

In then end, it seems likely that most of these protocols and systems have more to teach us than the other way around. Any protocol implemented by human beings, especially those evolved over time, is likely to be heavily optimized and carefully tuned against risk, whether by conscious analysis or *ad hoc* trial and error. How can we ignore them?

References

[AK96] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.

[AN96] Martín Abadi and Roger Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[And01] Ross Anderson. *Security Engineering*. Wiley, 2001.

[Arb99] W. A. Arbaugh. *Chaining Layered Integrity Checks*. PhD thesis, University of Pennsylvania, 1999.

[ASSW02] T. Anderson, S. Shenker, I. Stoica, and D. Wetherall. Design guidelines for robust internet protocols, 2002.

[BFIK99a] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The role of trust management in distributed systems security. In *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 185–210. Springer-Verlag Inc., New York, NY, USA, 1999.

[BFIK99b] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis. The KeyNote Trust Management System Version 2. Internet RFC 2704, September 1999.

[BFL96] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proc. of the 17th Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, Los Alamitos, 1996.

[BFN98] Matt Blaze, Joan Feigenbaum, and Moni Naor. A formal treatment of remotely keyed encryption. In *Proceedings of EUROCRYPT 98*, pages 251–265, 1998.

[BFS98] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance Checking in the PolicyMaker Trust-Management System. In *Proc. of Financial Cryptography '98, Lecture Notes in Computer Science, vol. 1465*, pages 254–274. Springer, Berlin, 1998.

[BIK03] M. Blaze, J. Ioannidis, and A. D. Keromytis. Experience with the KeyNote Trust Management System: Applications and Future Directions. In *Proceedings of the 1st International Conference on Trust Management*, pages 284–300, May 2003.

[Bla03] M. Blaze. Cryptology and physical security: Rights amplification in master-keyed mechanical locks. *IEEE Security and Privacy*, 1(2), March/April 2003.

[Com02] Federal Election Commission. Voting system standards, 2002.

[Ker01] A. D. Keromytis. *STRONGMAN: A Scalable Solution To Trust Management In Networks*. PhD thesis, University of Pennsylvania, 2001.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. *Proc. CRYPTO 99*, 1666:388–397, 1999.

[Koc96] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. *Proc. CRYPTO 96*, 1109:104–113, 1996.

[Kuh98] M. Kuhn. Cipher instruction search attack on the bus-encryption microcontroller ds50002fp. 47(10), October 1998.

[Kuh02] Markus G. Kuhn. Optical time-domain eavesdropping risks of crt displays. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 3–18, 2002.

[Nee94] R. M. Needham. Denial of service: An example. *Communications of the ACM*, 37(11):42–46, November 1994.

[Oeh97] M. Oehlert. *Safe Technican's Reference Manual*. AOLA, 1997.

[Tob00] M. W. Tobias. *Locks, Safes and Security*. C. Thomas, Ltd., 2000.