Preface

For generations, people have defined and protected their property and their privacy using locks, fences, signatures, seals, account books, and meters. These have been supported by a host of social constructs ranging from international treaties through national laws to manners and customs.

This is changing, and quickly. Most records are now electronic, from bank accounts to registers of real property; and transactions are increasingly electronic, as shopping moves to the Internet. Just as important, but less obvious, are the many everyday systems that have been quietly automated. Burglar alarms no longer wake up the neighborhood, but send silent messages to the police; students no longer fill their dormitory washers and dryers with coins, but credit them using a smartcard they recharge at the college bookstore; locks are no longer simple mechanical affairs, but are operated by electronic remote controls or swipe cards; and instead of renting videocassettes, millions of people get their movies from satellite or cable channels. Even the humble banknote is no longer just ink on paper, but may contain digital watermarks that enable many forgeries to be detected by machine.

How good is all this new security technology? Unfortunately, the honest answer is "nowhere near as good as it should be." New systems are often rapidly broken, and the same elementary mistakes are repeated in one application after another. It often takes four or five attempts to get a security design right, and that is far too many.

The media regularly report security breaches on the Internet; banks fight their customers over "phantom withdrawals" from cash machines; VISA reports huge increases in the number of disputed Internet credit card transactions; satellite TV companies hound pirates who copy their smartcards; and law enforcement agencies try to stake out territory in cyberspace with laws controlling the use of encryption. Worse still, features interact. A mobile phone that calls the last number again if one of the keys is pressed by accident may be just a minor nuisance—until someone invents a machine that dispenses a can of soft drink every time its phone number is called. When all of a sudden you find 50 cans of Coke on your phone bill, who is responsible, the phone company, the handset manufacturer, or the vending machine operator? Once almost every electronic device that affects your life is connected to the Internet—which Microsoft expects to happen by 2010—what does 'Internet security' mean to you, and how do you cope with it?

As well as the systems that fail, many systems just don't work well enough. Medical record systems don't let doctors share personal health information as they would like, but still don't protect it against inquisitive private eyes. Zillion-dollar military systems prevent anyone without a "top secret" clearance from getting at intelligence data, but are often designed so that almost everyone needs this clearance to do any work. Passenger ticket systems are designed to prevent customers cheating, but when trustbusters break up the railroad, they cannot stop the new rail companies cheating each other.

Many of these failures could have been foreseen if designers had just a little bit more knowledge of what had been tried, and had failed, elsewhere.

Security engineering is the new discipline that is starting to emerge out of all this chaos.

Although most of the underlying technologies (cryptology, software reliability, tamper resistance, security printing, auditing, etc.) are relatively well understood, the knowledge and experience of how to apply them effectively is much scarcer. And since the move from mechanical to digital mechanisms is happening everywhere at once, there just has not been time for the lessons learned to percolate through the engineering community. Time and again, we see the same old square wheels being reinvented.

The industries that have managed the transition most capably are often those that have been able to borrow an appropriate technology from another discipline. Examples include the reuse of technology designed for military identify-friend-or-foe equipment in bank cash machines and even prepayment gas meters. So even if a security designer has serious expertise in some particular speciality—whether as a mathematician working with ciphers or a chemist developing banknote inks—it is still prudent to have an overview of the whole subject. The essence of good security engineering is understanding the potential threats to a system, then applying an appropriate mix of protective measures—both technological and organizational—to control them. Knowing what has worked, and more importantly what has failed, in other applications is a great help in developing judgment. It can also save a lot of money.

The purpose of this book is to give a solid introduction to security engineering, as we understand it at the beginning of the twenty-first century. My goal is that it works at four different levels:

- *As a textbook that you can read from one end to the other over a few days as an introduction to the subject.* The book is to be used mainly by the working IT professional who needs to learn about the subject, but it can also be used in a one-semester course in a university.
- *As a reference book to which you can come for an overview of the workings of some particular type of system.* These systems include cash machines, taxi meters, radar jammers, anonymous medical record databases, and so on.
- *As an introduction to the underlying technologies, such as crypto, access control, inference control, tamper resistance, and seals.* Space prevents me from going into great depth; but I provide a basic road map for each subject, plus a reading list for the curious (and a list of open research problems for the prospective graduate student).
- *As an original scientific contribution in which I have tried to draw out the common principles that underlie security engineering, and the lessons that people building one kind of system should have learned from others.* In the many years I have been working in security, I keep coming across these. For example, a simple attack on stream ciphers wasn't known to the people who designed a common antiaircraft fire control radar so it was easy to jam; while a trick well known to the radar community wasn't understood by banknote printers and people who design copyright marking schemes, which led to a quite general attack on most digital watermarks.

I have tried to keep this book resolutely mid-Atlantic; a security engineering book has to be, as many of the fundamental technologies are American, while many of the interesting applications are European. (This isn't surprising given the better funding of U.S. universities and research labs, and the greater diversity of nations and markets in Europe.) What's more, many of the successful European innovations—from the smartcard to the GSM mobile phone to the pay-per-view TV service—have crossed the Atlantic and now thrive in the Americas. Both the science, and the case studies, are necessary.

This book grew out of the security engineering courses I teach at Cambridge University, but I have rewritten my notes to make them self-contained and added at least as much material again. It should be useful to the established professional security manager or consultant as a first-line reference; to the computer science professor doing research in cryptology; to the working police detective trying to figure out the latest computer scam; and to policy wonks struggling with the conflicts involved in regulating cryptography and anonymity. Above all, it is aimed at Dilbert. My main audience is the working programmer or engineer who is trying to design real systems that will keep on working despite the best efforts of customers, managers, and everybody else.

This book is divided into three parts.

- The first looks at basic concepts, starting with the central concept of a security protocol, and going on to human-computer interface issues, access controls, cryptology, and distributed system issues. It does not assume any particular technical background other than basic computer literacy. It is based on an Introduction to Security course that I teach to second-year undergraduates.
- The second part looks in much more detail at a number of important applications, such as military communications, medical record systems, cash machines, mobile phones, and pay-TV. These are used to introduce more of the advanced technologies and concepts. It also considers information security from the viewpoint of a number of different interest groups, such as companies, consumers, criminals, police, and spies. This material is drawn from my senior course on security, from research work, and from experience consulting.
- The third part looks at the organizational and policy issues: how computer security interacts with law, with evidence, and with corporate politics; how we can gain confidence that a system will perform as intended; and how the whole business of security engineering can best be managed.

I believe that building systems that continue to perform robustly in the face of malice is one of the most important, interesting, and difficult tasks facing engineers in the twenty-first century.

Ross Anderson
Cambridge, January 2001

About the Author

Why should I have been the person to write this book? Well, I seem to have accumulated the right mix of experience and qualifications over the last 25 years. I graduated in mathematics and natural science from Cambridge (England) in the 1970s, and got a qualification in computer engineering; my first proper job was in avionics; and I became interested in cryptology and computer security in the mid-1980s. After working in the banking industry for several years, I started doing consultancy for companies that designed equipment for banks, and then working on other applications of this technology, such as prepayment electricity meters.

I moved to academia in 1992, but continued to consult to industry on security technology. During the 1990s, the number of applications that employed cryptology rose rapidly: burglar alarms, car door locks, road toll tags, and satellite TV encryption systems all made their appearance. As the first legal disputes about these systems came along, I was lucky enough to be an expert witness in some of the important cases. The research team I lead had the good fortune to be in the right place at the right time when several crucial technologies, such as tamper resistance and digital watermarking, became hot topics.

By about 1996, it started to become clear to me that the existing textbooks were too specialized. The security textbooks focused on the access control mechanisms in operating systems, while the cryptology books gave very detailed expositions of the design of cryptographic algorithms and protocols. These topics are interesting, and important. However they are only part of the story. Most system designers are not overly concerned with crypto or operating system internals, but with how to use these tools effectively. They are quite right in this, as the inappropriate use of mechanisms is one of the main causes of security failure. I was encouraged by the success of a number of articles I wrote on security engineering (starting with "Why Cryptosystems Fail" in 1993); and the need to teach an undergraduate class in security led to the development of a set of lecture notes that made up about half of this book. Finally, in 1999, I got round to rewriting them for a general technical audience.

I have learned a lot in the process; writing down what you think you know is a good way of finding out what you don't. I have also had a lot of fun. I hope you have as much fun reading it!

Foreword

In a paper he wrote with Roger Needham, Ross Anderson coined the phrase "programming Satan's computer" to describe the problems faced by computer-security engineers. It's the sort of evocative image I've come to expect from Ross, and a phrase I've used ever since.

Programming a computer is straightforward: keep hammering away at the problem until the computer does what it's supposed to do. Large application programs and operating systems are a lot more complicated, but the methodology is basically the same. Writing a reliable computer program is much harder, because the program needs to work even in the face of random errors and mistakes: Murphy's computer, if you will. Significant research has gone into reliable software design, and there are many mission-critical software applications that are designed to withstand Murphy's Law.

Writing a *secure* computer program is another matter entirely. Security involves making sure things work, not in the presence of random faults, but in the face of an intelligent and malicious adversary trying to ensure that things fail in the worst possible way at the worst possible time ... again and again. It truly is programming Satan's computer.

Security engineering is different from any other kind of programming. It's a point I made over and over again: in my own book, *Secrets and Lies*, in my monthly newsletter *Crypto-Gram*, and in my other writings. And it's a point Ross makes in every chapter of this book. This is why, if you're doing any security engineering ... if you're even *thinking* of doing any security engineering, you need to read this book. It's the first, and only, end-to-end modern security design and engineering book ever written.

And it comes just in time. You can divide the history of the Internet into three waves. The first wave centered around mainframes and terminals. Computers were expensive and rare. The second wave, from about 1992 until now, centered around personal computers, browsers, and large application programs. And the third, starting now, will see the connection of all sorts of devices that are currently in proprietary networks, standalone, and non-computerized. By 2003, there will be more mobile phones connected to the Internet than computers. Within a few years we'll see many of the world's refrigerators, heart monitors, bus and train ticket dispensers, burglar alarms, and electricity meters talking IP. Personal computers will be a minority player on the Internet.

Security engineering, especially in this third wave, requires you to think differently. You need to figure out not how something works, but how something can be made to not work. You have to imagine an intelligent and malicious adversary inside your system (remember Satan's computer), constantly trying new ways to subvert it. You have to consider all the ways your system can fail, most of them having nothing to do with the design itself. You have to look at everything backwards, upside down, and sideways. You have to think like an alien.

As the late great science fiction editor John W. Campbell, said: "An alien thinks as well as a human, but not like a human." Computer security is a lot like that. Ross is

one of those rare people who can think like an alien, and then explain that thinking to humans. Have fun reading.

Bruce Schneier
January 2001

Acknowledgments

A great many people have helped in various ways with this book. I probably owe the greatest thanks to those who read the manuscript (or a large part of it) looking for errors and obscurities. They were Anne Anderson, Ian Brown, Nick Bohm, Richard Bondi, Caspar Bowden, Richard Clayton, Steve Early, Rich Graveman, Markus Kuhn, Dan Lough, David MacKay, John McHugh, Bob Morris, Roger Needham, Jerry Saltzer, Marv Schaefer, Karen Spärck Jones and Frank Stajano. Much credit also goes to my editor, Carol Long, who (among many other things) went through the first six chapters and coached me on the style appropriate for a professional (as opposed to academic) book. At the proofreading stage, I got quite invaluable help from Carola Bohm, Mike Bond, Richard Clayton, George Danezis, and Bruce Godfrey.

A large number of subject experts also helped me with particular chapters or sections. Richard Bondi helped me refine the definitions in Chapter 1; Jianxin Yan, Alan Blackwell and Alasdair Grant helped me investigate the applied psychology aspects of passwords; John Gordon and Sergei Skorobogatov were my main sources on remote key entry devices; Whit Diffie and Mike Brown on IFF; Steve Early on Unix security (although some of my material is based on lectures given by Ian Jackson); Mike Roe, Ian Kelly, Paul Leyland, and Fabien Petitcolas on the security of Windows NT4 and Win2K; Virgil Gligor on the history of memory overwriting attacks, and on mandatory integrity policies; and Jean Bacon on distributed systems. Gary Graunke told me the history of protection in Intel processors; Orr Dunkelman found many bugs in a draft of the crypto chapter and John Brazier pointed me to the Humpty Dumpty quote.

Moving to the second part of the book, the chapter on multilevel security was much improved by input from Jeremy Epstein, Virgil Gligor, Jong-Hyeon Lee, Ira Moskowitz, Paul Karger, Rick Smith, Frank Stajano, and Simon Wiseman, while Frank also helped with the following two chapters. The material on medical systems was originally developed with a number of people at the British Medical Association, most notably Fleur Fisher, Simon Jenkins, and Grant Kelly. Denise Schmandt-Besserat taught the world about bullae, which provided the background for the chapter on banking systems; that chapter was also strengthened by input from Fay Hider and Willie List. The chapter on alarms contains much that I was taught by Roger Needham, Peter Dean, John Martin, Frank Clish, and Gary Geldart. Nuclear command and control systems are much the brainchild of Gus Simmons; he and Bob Morris taught me much of what's in that chapter.

Sijbrand Spannenburg reviewed the chapter on security printing; and Roger Johnston has taught us all an enormous amount about seals. John Daugman helped polish the chapter on biometrics, as well as inventing iris scanning which I describe there. My tutors on tamper resistance were Oliver Kömmerling and Markus Kuhn; Markus also worked with me on emission security. I had substantial input on electronic warfare from Mike Brown and Owen Lewis. The chapter on phone fraud owes a lot to Duncan Campbell, Richard Cox, Rich Graveman, Udi Manber, Andrew Odlyzko and Roy Paterson. Ian Jackson contributed some ideas on network security. Fabien Petitcolas

‘wrote the book’ on copyright marking, and helped polish my chapter on it. Johann Bezuidenhoudt made perceptive comments on both phone fraud and electronic commerce, while Peter Landrock gave valuable input on bookkeeping and electronic commerce systems. Alistair Kelman was a fount of knowledge on the legal aspects of copyright; and Hal Varian kept me straight on matters of economics, and particularly the chapters on e-commerce and assurance.

As for the third part of the book, the chapter on e-policy was heavily influenced by colleagues at the Foundation for Information Policy Research, notably Caspar Bowden, Nick Bohm, Fleur Fisher, Brian Gladman, Ian Brown, Richard Clayton—and by the many others involved in the fight, including Whit Diffie, John Gilmore, Susan Landau, Brian Omotani and Mark Rotenberg. The chapter on management benefited from input from Robert Brady, Jack Lang, and Willie List. Finally, my thinking on assurance has been influenced by many people, including Robin Ball, Robert Brady, Willie List, and Robert Morris.

There were also many people over the years who taught me my trade. The foremost of them is Roger Needham, who was my thesis advisor; but I also learned a lot from hundreds of engineers, programmers, auditors, lawyers, and policemen with whom I worked on various consultancy jobs over the last 15 years. Of course, I take the rap for all the remaining errors and omissions.

Finally, I owe a huge debt to my family, especially to my wife Shireen for putting up with over a year in which I neglected household duties and was generally preoccupied. Daughter Bavani and dogs Jimmy, Bess, Belle, Hobbes, Bigfoot, Cat, and Dogmatix also had to compete for a diminished quantum of attention, and I thank them for their forbearance.

Legal Notice

I cannot emphasize too strongly that the tricks taught in this book are intended only to enable you to build better systems. They are not in any way given as a means of helping you to break into systems, subvert copyright protection mechanisms, or do anything else unethical or illegal.

Where possible I have tried to give case histories at a level of detail that illustrates the underlying principles without giving a "hacker's cookbook."

Should This Book Be Published at All?

There are people who believe that the knowledge contained in this book should not be published. This is an old debate; in previous centuries, people objected to the publication of books on locksmithing, on the grounds that they were likely to help the bad guys more than the good guys.

I think that these fears are answered in the first book in English that discussed cryptology. This was a treatise on optical and acoustic telegraphy written by Bishop John Wilkins in 1641 [805]. He traced scientific censorship back to the Egyptian priests who forbade the use of alphabetic writing on the grounds that it would spread literacy among the common people and thus foster dissent. As he said:

> *It will not follow that everything must be suppresst which may be abused... If all those useful inventions that are liable to abuse should therefore be concealed there is not any Art or Science which may be lawfully profest.*

The question was raised again in the nineteenth century, when some well-meaning people wanted to ban books on locksmithing. A contemporary writer on the subject replied [750]:

> *Many well-meaning persons suppose that the discussion respecting the means for baffling the supposed safety of locks offers a premium for dishonesty, by showing others how to be dishonest. This is a fallacy. Rogues are very keen in their profession, and already know much more than we can teach them respecting their several kinds of roguery. Rogues knew a good deal about lockpicking long before locksmiths discussed it among themselves ... if there be harm, it will be much more than counterbalanced by good.*

These views have been borne out by long experience since. As for me, I worked for two separate banks for three and a half years on cash machine security, but I learned significant new tricks from a document written by a convicted card fraudster that circulated in the U.K. prison system. Many government agencies are now coming round to this point of view. It is encouraging to see, for example, that the U.S. National Security Agency has published the specifications of the encryption algorithm (Skipjack) and the key management protocol (KEA) used to protect secret U.S. government traffic.

Their judgment is clearly that the potential harm done by letting the Iraqis use a decent encryption algorithm is less than the good that will be done by having commercial off-the-shelf software compatible with Federal encryption standards.

In short, while some bad guys will benefit from a book such as this, they mostly know the tricks already, and the good guys will benefit much more.