Preface to the Third Edition

The first edition of *Security Engineering* was published in 2001 and the second in 2008. Since then there have been huge changes.

The most obvious is that the smartphone has displaced the PC and laptop. Most of the world's population now walk around with a computer that's also a phone, a camera and a satnav; and the apps that run on these magic devices have displaced many of the things we were building ten years ago. Taxi rides are now charged by ride-hailing apps rather than by taxi meters. Banking has largely gone online, with phones starting to displace credit cards. Energy saving is no longer about your meter talking to your heating system but about both talking to your phone. Social networking has taken over many people's lives, driving everything from advertising to politics.

A related but less visible change is the move to large server farms. Sensitive data have moved from servers in schools, doctors' offices and law firms to cloud service providers. Many people no longer do their writing on word processing software on their laptop but on Google Docs or Office365 (I'm writing this book on Overleaf). This has consequences. Security breaches can happen at a scale no-one would have imagined twenty years ago. Compromises of tens of millions of passwords, or credit cards, have become almost routine. And in 2013, we discovered that fifteen years' worth of UK hospital medical records had been sold to 1200 organisations worldwide without the consent of the patients (who were still identifable via their postcodes and dates of birth).

A real game-changer of the last decade was the Snowden revelations, also in 2013, when over 50,000 Top Secret documents about the NSA's signals intelligence activities were leaked to the press. The scale and intrusiveness of government surveillance surprised even cynical security engineers. It followed on from Stuxnet, where America attacked Iran's nuclear weapons program using malware, and was followed by NotPetya, where a Russian

cyberweapon, deployed against the Ukraine, inflicted hundreds of millions of dollars' worth of collateral damage on firms elsewhere. This brings us to the third big change, which is a much better understanding of nation-state security threats. In addition to understanding the capabilities and priorities of western intelligence agencies, we have a reasonably good idea of what the Chinese, the Russians and even the Syrians get up to.

And where the money is, the crooks follow too. The last decade has also seen the emergence of a cyber-crime ecosystem, with malware writers providing the tools to subvert millions of machines, many of which are used as criminal infrastructure while others are subverted in various ways into defrauding their users. We have a team at Cambridge that studies this, and so do dozens of other research groups worldwide. The rise of cybercrime is changing policing, and other state activity too: cryptocurrencies are not just making it easier to write ransomware, but undermining financial regulation. And then there are non-financial threats from cyber-bullying up through hate speech to election manipulation and videos of rape and murder.

So online harms now engage all sorts of people from teachers and the police to banks and the military. It is ever more important to measure the costs of these harms, and the effectiveness of the measures we deploy to mitigate them.

Some of the changes would have really surprised someone who read my book ten years ago and then spent a decade in solitary confinement. For example, the multilevel security industry is moribund, despite being the beneficiary of billions of dollars of US government funding over forty years; the Pentagon's entire information security philosophy – of mandating architectures to stop information flowing downward from Top Secret to Secret to Confidential to Unclassified – has been abandoned as unworkable. While architecture still matters, the emphasis has shifted to ecosystems. Given that bugs are ubiquitous and exploits inevitable, we had better be good at detecting exploits, fixing bugs and recovering from attacks. The game is no longer trusted systems but coordinated disclosure, DevSecOps and resilience.

What might the future hold? A likely game-changer is that as we put software into safety-critical systems like cars and medical devices, and connect them to the Internet, safety and security engineering are converging. This is leading to real strains; while security engineers fix bugs quickly, safety engineers like to test systems rigorously against standards that change slowly if at all. A wicked problem is how we will patch durable goods. At present, you might get security patches for your phone for three years and your laptop for five; you're expected to buy a new one after that. But cars last for fifteen years on average and if we're suddenly asked to scrap them after five the environmental costs won't be acceptable. So tell me, if you're writing navigation software today in 2020 for a car that will launch in 2023, how will you ensure that you can keep on shipping security patches in 2033, 2043 and 2053? What tools will you choose today?

Finally, there has been a sea change in the political environment. After decades in which political leaders considered technology policy to be for men in anoraks, and generally took the line of least resistance, the reports of Russian interference in the Brexit referendum and the Trump election got their attention. The prospect of losing your job can concentrate the mind wonderfully. The close attention of lawmakers is changing the game, first with tighter general rules such as Europe's General Data Protection Regulation; and second as products that are already regulated for safety, from cars and railway signals to children's toys acquire software and online connectivity, which has led to rules in Europe about how long software has to be maintained.

The questions the security engineer has to ask today are just the same as a decade ago: what are we seeking to prevent, and will the proposed mechanisms actually work? However, the canvas on which we work is now much broader. Almost all human life is there.

Ross Anderson
Cambridge, October 2020

Preface to the Second Edition

The first edition of *Security Engineering* was published in May 2001. Since then the world has changed.

System security was one of Microsoft's lowest priorities then; it's now one of the highest. The volume of malware continues to increase along with the nuisance that it causes. Although a lot of effort has gone into defence – we have seen Windows NT replaced by XP and then Vista, and occasional service packs replaced by monthly security patches – the effort put into attacks has increased far more. People who write viruses no longer do so for fun, but for profit; the last few years have seen the emergence of a criminal economy that supports diverse specialists. Spammers, virus writers, phishermen, money launderers and spies trade busily with each other.

Cryptography has also moved on. The Advanced Encryption Standard is being embedded into more and more products, and we have some interesting developments on the public-key side of things too. But just as our algorithm problems get solved, so we face a host of implementation issues. Side channels, poorly designed APIs and protocol failures continue to break systems. Applied cryptography is harder than ever to do well.

Pervasive computing also opens up new challenges. As computers and communications become embedded invisibly everywhere, so problems that used to only afflict 'proper computers' crop up in all sorts of other devices too. What does it mean for a thermometer to be secure, or an air-conditioner?

The great diversity of intelligent devices brings with it a great diversity of interests and actors. Security is not just about keeping the bad guys out, but increasingly concerned with tussles for power and control. DRM pits the content and platform industries against consumers, and against each other; accessory control is used to tie printers to their vendors' cartridges, but leads to antitrust lawsuits and government intervention. Security also interacts with

safety in applications from cars through utilities to electronic healthcare. The security engineer needs to understand not just crypto and operating systems, but economics and human factors as well.

And the ubiquity of digital devices means that 'computer security' is no longer just a problem for a few systems specialists. Almost all white-collar crime (and much crime of the serious violent sort) now involves computers or mobile phones, so a detective needs to understand computer forensics just as she needs to know how to drive. More and more lawyers, accountants, managers and other people with no formal engineering training are going to have to understand system security in order to do their jobs well.

The rapid growth of online services, from Google and Facebook to massively multiplayer games, has also changed the world. Bugs in online applications can be fixed rapidly once they're noticed, but the applications get ever more complex and their side-effects harder to predict. We may have a reasonably good idea what it means for an operating system or even a banking service to be secure, but we can't make any such claims for online lifestyles that evolve all the time. We're entering a novel world of evolving socio-technical systems, and that raises profound questions about how the evolution is driven and who is in control.

The largest changes, however, may be those driven by the tragic events of September 2001 and by our reaction to them. These have altered perceptions and priorities in many ways, and changed the shape of the security industry. Terrorism is not just about risk, but about the perception of risk, and about the manipulation of perception. This adds psychology and politics to the mix. Security engineers also have a duty to contribute to the political debate. Where inappropriate reactions to terrorist crimes have led to major waste of resources and unforced policy errors, we have to keep on educating people to ask a few simple questions: what are we seeking to prevent, and will the proposed mechanisms actually work?

Ross Anderson
Cambridge, January 2008

Preface to the First Edition

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:

1. 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;
2. as a reference book to which you can come for an overview of the workings of some particular type of system (such as cash machines, taxi meters, radar jammers, anonymous medical record databases or whatever);
3. 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);
4. 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 anti-aircraft 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 US 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 the human-computer interface, 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 which we 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, the 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 which 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

For my daughter, and other lawyers …

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 or do anything else illegal. So 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'.

Governments fought to restrict knowledge of cryptography until the turn of the century, and there may still be people who believe that the knowledge contained in this book should not be published.

Their fears were answered in the first book in English that discussed cryptology, a 1641 treatise on optical and acoustic telegraphy written by Oliver Cromwell's cryptographer and son-in-law John Wilkins [2025]. 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 of 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. In 1853, a contemporary writer replied [1899]:

> *'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.'

Thirty years later, in the first book on cryptographic engineering, Auguste Kerckhoffs explained that you must always assume that the other side knows the system, so security must reside in the choice of a key.

His wisdom has been borne out by long experience since. The relative benefits of 'Open' versus 'Closed' security systems have also been studied by researchers applying the tools of dependability analysis and security economics. We discuss their findings in this book.

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

Ross Anderson
Cambridge, November 2020

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