**Chapter 24**

**Copyright and DRM**

**Be very glad that your PC is insecure – it means that after you buy**

**it, you can break into it and install whatever software you want.**

**What YOU want, not what Sony or Warner or AOL wants.**

– JOHN GILMORE

**24.1** **Introduction**

Copyright has been among the highly contentious issues of the digital age, and  
 drove the development of *digital rights management* (DRM). The big ﬁght was  
 between Hollywood and the tech industry in the 1990s and 2000s; by 2010 it  
 had essentially been resolved. We won; power in the music and ﬁlm industry  
 passed from ﬁrms like EMI and Universal to ﬁrms like Apple, Spotify, Amazon  
 and Netﬂix, while Amazon cornered the market in books – ﬁrst physically and  
 then with e-books. Technically, the world moved from enjoying music and video  
 from local media such as CDs and DVDs (which many people used to share)  
 and satellite broadcast TV (which some people used to hack), to broadband  
 streaming services where subscription management is fairly straightforward. I  
 thought seriously about dropping this chapter from the third edition and just  
 referring you to the second edition chapter online, as there’s not a lot more to  
 say technically. On reﬂection I decided to edit it to give the context as seen  
 from 2020. Just as the multilevel secure systems I describe in Chapter 9 are  
 largely obsolete but drove the development of military computer security and  
 inﬂuenced today’s security landscape in many subtle ways, so also the copyright  
 wars left their mark. DRM is still used: in ebooks, in the Fairplay system on  
 your iPhone to make it harder to copy songs, and in HTML5 in your browser to  
 make it harder for you to copy Netﬂix videos. Very similar techniques are used  
 in gaming platforms to make it harder for players to use aimbots, in protecting  
 user data on cloud platforms, and in mobile phone security where *Runtime*  
 *Application Self-Protection* (RASP) is used to defend banking and other apps  
 against malware that roots the phone. Accessory-control mechanisms that our  
 industry adopted to protect game cartridges now use cryptography to support  
 business models in dozens of business sectors. My ﬁnal reason to spare this

737

*24.1. INTRODUCTION*

chapter is that the copyright wars became part of our shared security culture,  
 and even if you’re too young to have taken part, you may occasionally ﬁnd it  
 helpful to understand what we greybeards are blethering on about.

At the political level, the control of information has been near the centre

of government concerns since before William Tyndale (one of the founders of  
 the Cambridge University Press) was burned at the stake for printing the Bible  
 in English. The sensitivity continued through the establishment of modern

copyright law starting with the Statute of Anne in 1709, through eighteenth-  
 century battles over press censorship, to the Enlightenment and the framing of  
 the US Constitution. The link between copyright and censorship is obscured  
 by technology from time to time, but has a habit of reappearing. Copyright  
 mechanisms exist to keep information out of the hands of people who haven’t  
 paid for it, while censors keep information out of the hands of people who aren’t  
 trusted with it. Where ISPs are compelled to install ﬁlters that prevent their  
 customers from downloading copyrighted material, these ﬁlters can often be  
 used to block seditious material too.

Over the twentieth century, the great wealth accruing to the owners of lit-

erary copyright, ﬁlms and music created a powerful interest in control. As the  
 Internet took off, the music and ﬁlm industries feared losing sales to digital  
 copying, and lobbied for sweetheart laws – the DMCA in America in 1998, and  
 a series of IP Directives in Europe – that give special legal protection to mech-  
 anisms that enforce copyright. These laws have since been used and abused  
 for all sorts of other purposes, from taking down phishing websites to stopping  
 people from reﬁlling printer cartridges and even from repairing broken devices.

The ostensible target of these laws was the DRM used from the 1990s in

products such as Windows Media Player, and since 2017 in browsers compliant  
 with HTML5, to control the copying of music and videos. The basic idea in DRM  
 is to make a ﬁle uncopiable by encrypting it, and then providing separately a  
 ‘license’ which is the key to the media ﬁle encrypted using a key unique to the  
 user, plus some statements in a ‘rights management language’ about what the  
 user can do with the content. The app that renders the media content is trusted  
 to abide by these. I’ll also give a quick tour of the history and describe some  
 interesting variants such as satellite TV encryption systems, copyright marking  
 and traitor tracing. DRM is less relevant now than in 2008 when the second  
 edition of this book came out, but there are still some applications, which I’ll  
 describe later.

Some serious policy issues are mixed up in all this. It’s hard to make DRM

compatible with open-source software unless you have either trustworthy hard-  
 ware such as enclaves or TPMs, or closed-source sandboxes that are patched as  
 soon as they are reverse engineered. The computer industry resisted DRM but  
 Hollywood and the music industry forced us to introduce it, saying that without  
 it they’d be ruined. We warned them that DRM would ruin them, and they  
 didn’t listen. Music is no longer run by ﬁrms like Universal and EMI but by  
 ﬁrms like Apple and Amazon – and the move to streaming let new ﬁrms like  
 Spotify join the party. DRM introduced serious privacy issues, though, which  
 have not gone away with streaming. Instead of a license management server in  
 Microsoft knowing every music track you’ve ever listened to, and every movie  
 you’ve ever watched, it’s now streaming servers at Apple or Spotify or Netﬂix.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 738 | Ross Anderson |

*24.2. COPYRIGHT*

**24.2** **Copyright**

The protection of copyright has for years been an obsession of the ﬁlm, music  
 and book publishing industries. There were long and acrimonious disputes in  
 many countries about whether blank audiocasettes, and then videocassettes,  
 should be subjected to a tax whose proceeds would be distributed to copyright  
 owners. Going back to the nineteenth century, there was alarm that the inven-  
 tion of photography would destroy the book publishing trade; the eighteenth  
 saw book publishers trying to close down public lending libraries, until they re-  
 alised they were creating mass literacy and driving sales; while in the sixteenth,  
 the invention of movable type printing was considered subversive by most of the  
 powers of the day, from princes and bishops to craft guilds.

We’ll come back to these historical examples later. But I’m going to start

by looking at software protection – as most of the copyright issues that led to  
 DRM played out in the PC and games software markets from the 1980s.

**24.2.1** **Software**

Software for early computers was given away free by the hardware vendors or  
 by users who’d written it. IBM even set up a scheme in the 1960s whereby its  
 users could share programs they’d written. (Most business programs were too  
 specialised, too poorly documented, or just too hard to adapt. But software  
 used in research was widely shared.) So protecting software copyright was not  
 an issue. Almost all organizations that owned computers were large and re-  
 spectable; their software tended to require skilled maintenance. There were also  
 computer bureau services – the forerunner of today’s cloud computing – where  
 the owner of a mainframe who used it to work out their own payroll would offer  
 this as a service to other ﬁrms. There, you bought the service, not the software.  
 The hardware costs were the dominant factor.

When minicomputers arrived in the 1960s, software costs became signiﬁcant.

Hardware vendors started to charge extra for their operating system, and third-  
 party system houses sprang up. To begin with, they mostly sold you a complete  
 bespoke system – hardware, software and maintenance – so piracy was still not  
 much of an issue. By the mid-1970s, some of them had turned bespoke systems  
 into packages: software originally written for one bakery would be parametrised  
 and sold to many bakeries. The most common copyright dispute in those days  
 was when a programmer left your company to join a competitor, and their code  
 suddenly acquired a number of your features; the question then was whether  
 he’d taken code with him, or reimplemented it.

One way to resolve such a problem is to look at *software birthmarks* – fea-

tures of how a particular implementation was done. For example, litigation over  
 whether people had copied software from the ROM of the early IBM PCs turned  
 on the order in which registers are pushed and popped, as the software had been  
 written in assembler. This merged with the ﬁeld of *stylometry* in which human-  
 ities scholars try to attribute authorship by analysis of writing styles1. More

1The cryptanalyst William Friedman and his wife Elizebeth were hired by an eccen-

tric millionaire to ﬁgure out whether Bacon wrote Shakespeare. They concluded that he

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 739 | Ross Anderson |

*24.2. COPYRIGHT*

recently, the natural-language processing community has written plagiarism de-  
 tection tools, which typically recognise a passage of text by indexing it according  
 to the least common words that appear in it [879]; by the 1990s this had led to  
 tools that try to identify malware authors from their coding style [1099]. Code  
 stylometry is still an active area of research [370].

With time, people invented lots of useful things to do with software. So a

ﬁrm that had bought a minicomputer for stock control (or contracted for time  
 on a bureau service) might be tempted to run a statistical program as well to  
 prepare management reports. Meanwhile, the installed base of machines got  
 large enough for software sharing to happen more than just occasionally. So  
 some system houses started to design enforcement mechanisms. A common one  
 was to check the processor serial number; another was the *time bomb*. When I  
 worked in 1981 for a company selling retail stock control systems, we caused a  
 message to come up every few months saying something like “Fault no. WXYZ  
 – please call technical support”. WXYZ was an encrypted version of the license  
 serial number, and if the caller claimed to be from that customer we’d give them  
 a password to re-enable the system for the next few months. (If not, we’d send  
 round a sales person.) This mechanism could have been defeated easily if the  
 ‘customer’ understood it, but in practice it worked ﬁne: most of the time it was  
 a low-level clerk who got the fault message and called our office.

Software copyright infringement really started to become an issue when the

arrival of microcomputers in the late 1970s and early 80s created a mass market,  
 and software houses started to ship products that didn’t need technical support  
 to install and run. Initial responses varied. There was a famous open letter from  
 Bill Gates in 1976, a year after Microsoft was founded, in which he complained  
 that less than 10% of all microcomputer users had paid them for BASIC [722].  
 “Who cares if the people who worked on it get paid?” he asked. “Is this fair?”  
 His letter concluded: “Nothing would please me more than being able to hire  
 ten programmers and deluge the hobby market with good software.”

Appeals to fair play only got so far, and the industry next tackled the main

difference between minis and the early micros – the latter had no processor serial  
 numbers. There were three general approaches tried: to add uniqueness on to  
 the machine, to create uniqueness in it, or to use whatever uniqueness happened  
 to exist already by chance.

1. The standard way to add hardware uniqueness was a *dongle* – a device

attached to the PC which could be interrogated by the software. The

simplest just had a serial number; the most common executed a sim-  
 ple challenge-response protocol; while some top-end devices actually per-  
 formed some critical part of the computation.

2. A very common strategy in the early days was for the software to install

itself on a PC’s hard disk in a way that resisted naive copying. For exam-  
 ple, a sector of the hard disk would be marked as bad, and a critical part  
 of the code or data written there. Now if the product were copied from the  
 hard disk using the standard utilities, the bad sector wouldn’t be copied  
 and the copy wouldn’t work. A variant on the same theme was to require

hadn’t. [1001].

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 740 | Ross Anderson |

*24.2. COPYRIGHT*

the presence of a master diskette which had been customized in some way,  
 such as by formatting it in a strange way or even burning holes in it with  
 a laser. In general, though, a distinction should be drawn between pro-  
 tecting the copy and protecting the master; it’s often a requirement that  
 people should be able to make copies for backup if they wish, but not to  
 make copies of the copies (this is called *copy generation control*).

3. 1988 saw the arrival of the *license server*, basically a machine programmed

to act as a dongle shared by all the machines on a company network, which  
 supported more complex business models such as enabling a company to  
 buy the right to run a program on up to 20 machines at once, and enabling  
 multiple software companies to license their products via the same license  
 server.

4. A product I worked on in 1989 ﬁngerprinted the PC – what extension

cards were present, how much memory, what type of printer – and if this  
 conﬁguration changed too radically, it would ask the user to phone the  
 helpline. It’s quite surprising how many unique identiﬁers there are in the  
 average PC; ethernet addresses and serial numbers of disk controllers are  
 only the more obvious ones. So you can tie software to a given machine  
 ﬁngerprint; ad trackers use similar techniques to this day.

A generic attack that works against most of these defenses is to go through

the software with a debugger and remove all the calls made to the copy protec-  
 tion routines. Many hobbyists did this for sport, and competed to put unpro-  
 tected versions of software products online as soon as possible after their launch.  
 Even people with licensed copies of the software often got hold of unprotected  
 versions as they were easier to back up and often more reliable generally. You  
 can stop this by having critical code somewhere uncopiable (such as in a dongle,  
 a license server, or nowadays in the cloud) but this arms race taught everyone  
 that if you don’t do something like that then kids with debuggers will always  
 break your scheme eventually. It’s one reason why closed platforms, like games  
 consoles and the iPhone, only run signed code.

The vendors also used psychological techniques.

*•* The installation routine for many business programs would embed the  
 toolbar. This wouldn’t stop a pirate distributing copies registered in a  
 false name, but it will discourage legitimate users from giving casual copies  
 to colleagues. To this day, when I download papers from many academic  
 journals, my university’s name and a serial number are visible in the pdf.  
 These are examples of *copyright marking* which I’ll discuss in more detail  
 later.

*•* Industry people delighted in telling tales of organizations that had come

*•* If early Microsoft software (Multiplan, Word or Chart) thought you were  
 evil bears bitter fruit. Now trashing program disk.’ It would then seek to  
 track zero on the ﬂoppy disk and go ‘rrnt, rrnt, rrnt’.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 741 | Ross Anderson |

*24.2. COPYRIGHT*

In the late-1980s, the market split. The games market moved to hardware

protection, and ended up dominated by consoles with closed architectures whose  
 software was sold in proprietary cartridges. As consumers are more sensitive  
 about the sticker price of a product than about its total cost of ownership, it  
 makes sense to subsidise the console out of later sales of software. This led to  
 *accessory control* in which hardware protection is used to control aftermarkets;  
 it was adopted by ﬁrms selling printers and much else. We’ll discuss it in detail  
 in section 24.6.

Business software vendors moved from dongles to license servers for high-

value products such as the CAD software used to design everything from chips  
 to ships. Technical support is often critical for such products, so they may be  
 sold as a bundle of software and service. But vendors generally stopped trying  
 to protect mass-market products using technical means, for several reasons.

*•* Unless you’re prepared to spend money on dongle hardware to execute  
 defeated by people for whom it’s an intellectual challenge, and unprotected  
 code will be published anonymously.

*•* Protection was a nuisance. Multiple dongles get in the way or interfere  
 and recovery; they also cause software from different vendors to be incom-  
 patible and in some cases unable to reside on the same machine. (The  
 difficulty of doing this right is one reason why so many of the ﬁrms who  
 use license management use Flexlm.)

*•* Many vendors preferred not to have to worry about whether the soft-  
 machine) or to the machine (in which case he could sell the computer  
 second-hand with the software installed). As both practices were com-  
 mon, mechanisms that made one or the other very much harder caused  
 problems. Mechanisms that could deal with both (such as dongles and  
 license servers) tended to be expensive.

*•* The arrival of computer viruses forced corporate customers to invest in  
 a few years, antivirus programs made life much harder for copy protection  
 mechanisms in any case, as non-standard operating system usage tended  
 to set off alarms.

*•* There was not much money to be made out of harassing personal users as  
 rather than pay.

*•* A certain level of sharing was good for business. People who got a pirate  
 their employer to buy one. In 1998 Bill Gates even said, “Although about  
 three million computers get sold every year in China, people don’t pay for  
 the software. Someday they will, though. And as long as they’re going to  
 steal it, we want them to steal ours. They’ll get sort of addicted, and then  
 we’ll somehow ﬁgure out how to collect sometime in the next decade”[755].

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 742 | Ross Anderson |

*24.2. COPYRIGHT*

*•* Competition led to falling costs which made piracy less attractive. In the  
 Turbo Pascal in 1983. Before then a typical language compiler cost about  
 $500 and came with such poor documentation that you had to spend a  
 further $50 on a book to tell you how to use it. Borland’s product cost  
 $49.95, was technically superior to Microsoft’s, and came with a manual  
 that was just as good as a third party product. (So, like many other

people, once I’d heard of it, borrowed a copy from a friend, tried it and  
 liked it, I went out and bought it.) ‘Pile it high and sell it cheap’ simply  
 proved to be a more proﬁtable business model.

The industry then turned to the law. Software is mostly protected by copy-

right law; when you write software (or a book, or a tune) copyright comes  
 into existence automatically nowadays and you have the right to sue people  
 for damages if they make copies without your permission. The details vary by  
 country but copyright infringement tends to be a crime only if done at commer-  
 cial scale. So copyright owners can send unpleasant letters to individuals and  
 small businesses, but actually suing them for a few dollars or pounds or euros in  
 the small claims court is uneconomic. Against large-scale users, though, copy-  
 right enforcement can be worthwhile. In fact, when IBM separated its hardware  
 and software businesses in 1969 – following a lawsuit from the US government  
 which claimed that bundling software with hardware entrenched their market  
 dominance – they took a strategic decision not to use any technical copyright en-  
 forcement mechanisms as they would be onerous to customers and not effective  
 against clever thieves, so they’d rely on the law instead [1793].

In 1988, Microsoft led the industry in IBM’s footsteps, and established trade

organizations (such as the Business Software Alliance in the USA) that brought  
 high-proﬁle prosecutions of large companies that had been condoning widespread  
 use of unlicensed software. This was followed up by harassing medium and even  
 small businesses with threatening letters demanding details of the company’s  
 policy on enforcing copyright – basically demanding they sign up for an approved  
 software audit schemes or risk a raid by an enforcement squad.

The industry discovered that the law not only provides tools for enforce-

ment, but sets limits too. In 1993, a software company director in Scunthorpe,  
 England, received a criminal conviction under Britain’s Computer Misuse Act  
 for ‘making an unauthorized modiﬁcation’ to a system. Their customers had to  
 enter unlock codes regularly into his software or it froze, denying access to data.  
 But when he used this mechanism to enforce payment of a disputed invoice, the  
 court decided he’d gone too far, and he ended up with a criminal record [455].

Thanks to the ubiquity of Office, Microsoft had by then become a tax on the

corporate sector, making most of its revenue from customers with over 25,000  
 licenses. In addition to Office, it was selling many high-value products for

network management and other tasks, so like the CAD ﬁrms it turned to license  
 servers. Although these could still be defeated by disassembling the application  
 code, this got harder as code became larger, and was unattractive to large ﬁrms  
 after a few of them had been sued. Then the very idea of running on unlicensed  
 software became crazy when Patch Tuesday arrived in 2003. With personal

software, the emphasis shifted to online registration: you’d design your product

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 743 | Ross Anderson |

*24.2. COPYRIGHT*

to get customers to interact with your web site – whether to download the tunes,  
 latest exchange rates or security updates. Large-scale commercial counterfeiting  
 can then be detected by monitoring product serial numbers registered online2.

I wrote in the second edition of this book in 2008: “software-as-a-service may

be the ultimate copyright protection or DRM for software (or any other content  
 that can live online): you can’t buy it, freeze the version you’re running, or use  
 it offline. You may also get to control all your customers’ data too, giving you  
 impressive lockin”. That is precisely the model to which the software industry  
 has converged since the early 2010s. Putting some or all of the functionality in  
 the cloud can give real advantages of cost and reliability, which I will discuss  
 in section 27.5.5. Software is then sold by subscription and the issue of copy  
 protection goes away.

**24.2.2** **Free software, free culture?**

In the old days, software was shared and this continued to be the case among aca-  
 demics and other research scientists, who evolved many communities of practice  
 within which software was shared freely and adapted by successive contributors.  
 This continued to support the dominant platforms of the time, which initially  
 meant IBM. During the 1970s, for example, the UK government pushed British  
 academics to buy ICL computers; ICL was Britain’s champion, having been set  
 up in the 1960s when the government nationalised the computer industry to  
 ‘save’ it from IBM. However, we academics wanted IBM mainframes as other  
 academics worldwide had written software that ran on their hardware, and even  
 although most was written in high-level languages like FORTRAN, porting it  
 was a hassle. The arrival of home computers in the 1970s and the PC in 1981  
 developed ever wider communities of software enthusiasts who shared our work,  
 whether by physically passing diskettes around friends or in clubs, or via early  
 bulletin-board systems and other dial-up networks.

In 1983 IBM stopped supplying the source code for its products, introducing

a policy of ‘object code only’, and other vendors followed. This made it a

lot harder to understand the platforms and tools on which we relied and led to  
 pushback on a number of fronts. Two years later, Richard Stallman, an engineer  
 at MIT, was annoyed when he could not integrate a new Xerox printer with the  
 local maintenance arrangements as Xerox would not supply source code for the  
 printer driver. He announced the GNU project to build a free operating system,  
 and helped found the Free Software Foundation (FSF), which promoted the  
 idea of free software. Free software means that users should be able to run it for  
 any purpose, study how it works and change it, and redistribute it – including  
 improved or modiﬁed versions. One slogan was ‘free as in free speech, not as  
 in free beer’, but free software comes in many ﬂavours. The FSF promoted  
 the GNU General Public License (GPL) which has the property that anyone  
 adapting GPL licensed software and making it available must make the source

2Once they got product registration sorted out, Microsoft found that a third of the copies

of Office sold in Germany were counterfeit, and traced them to a small factory a few miles up  
 the road from us in Cambridge. Almost all the factory’s staff were unaware of the scam – they  
 believed the company was a bona ﬁde Microsoft supplier. They were proud of their product  
 and their sales staff used it to try to get CD duplication business from other software houses.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 744 | Ross Anderson |

*24.2. COPYRIGHT*

code of their adaptation publicly available, under the same license – a viral  
 property also known as ‘copyleft’. In 1988, the University of California released  
 the Berkeley distribution of Unix under the less restrictive BSD license that  
 simply allows anyone to use the software for any purpose.

Such licensing arrangements are necessary because otherwise an operating

system that had been written by 500 different people over 20 years would contain  
 code that was their copyright, and so any of them could go to court to exercise  
 their right to prevent some third party from using it. Proprietary software

vendors can get the copyright in code written by engineers they employ3, but  
 what about projects maintained by volunteers? Open licenses help avoid thickets  
 of conﬂicting claims.

There was much argument through the 1990s about their respective merits,

but both approaches are in wide use. Linux was ﬁrst released in 1991 under  
 the GPL, while Berkeley Unix spawned FreeBSD and other variants which are  
 available under the BSD license. As we noted in the chapter on Access Controls,  
 Linux was the platform on which Android was built, while FreeBSD evolved into  
 OSX and iOS. Other free software licenses were developed for Apache and in  
 other communities, and public licenses spread quickly from software to other  
 creative activities: for example, a variant of BSD was adapted for Wikipedia.

Software and culture both involve the adaptive and cumulative contributions

of many individuals. Traditional musicians sometimes compose new tunes but  
 more often change existing ones; even new compositions draw on phrases from  
 the existing vocabulary. DJs rip tracks from others and mash them together  
 into new compositions. Novelists reuse old storylines and character stereotypes,  
 while comedians recycle old jokes. The law doesn’t always deal with this very  
 well as it tends to be written for large corporate interests rather than for com-  
 munities. So music companies would press musicians to write entirely new tunes  
 with clean copyrights rather than following tradition and adapting the best tunes  
 of the older players.

Academia is also a place where we build on each others’ work, and has the

further twist that we get our recognition from the number of people who use our  
 work rather than the number of people who pay for it. Mathematicians become  
 famous if lots of other mathematicians use their theorems in other results, and  
 computer scientists get recognition if lots of people use our software. This creates  
 real tensions with publishers. Indeed, starting in the 1970s, many computer  
 scientists made both our code and our publications available freely online, using  
 FTP servers and later, once they were invented, web pages. We tended to

ignore the copyright agreements we had to sign with academic journals to get  
 our papers published – or if we were careful we crossed out the ‘exclusive’ clause  
 in the agreements, which back then were paper forms that the publishers never  
 bothered to check.

1994 saw a couple of publications with real impact. Andrew Odlyzko calcu-

lated that the U.S. government spent about $100M a year doing mathematics

3The law varies from one country to another. In some countries, such as the USA, you

own copyright in a program written by an employee, while in others you have to make it a  
 term in an employment contract; and contractors are another matter altogether. And since  
 the pandemic lockdown, half my team are working from home in different countries. It really  
 is prudent to have a written agreement.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 745 | Ross Anderson |

*24.2. COPYRIGHT*

(by paying professors’ salaries and the stipends of grad students) and a fur-  
 ther $100M a year marketing mathematics (being the money that was spent in  
 journals and conferences, plus the unpaid labour that mathematicians put in  
 so journal publishers could make their proﬁts) [1459]. If publication went fully  
 online and all papers were available for all to read, perhaps the amount spent  
 on actual mathematics could be increased. A quarter of a century and many  
 tussles later, most government and charitable funders insist that the research  
 they pay for is made available to all (though the journals have survived very  
 comfortably by imposing page charges on authors, and also demanding that  
 university libraries buy subscriptions for online access to their back catalogue).

The second, and better-known, was a paper by EFF founder John Perry

Barlow, who was also a lyricist for the Grateful Dead. He pointed out that as  
 the marginal cost of copying is zero with digital technology, ‘information wants  
 to be free’ (which he ascribed to Stewart Brand). Both the physical containers of  
 ideas (books, CDs) were vanishing, as was jurisdiction, as the Internet enabled  
 people to swap ﬁles across national boundaries. He warned against corporate  
 legal departments trying to protect by force what could no longer be protected  
 by practical efficiency or general social consent, and about the USA writing  
 copyright compliance into trade treaties: “Ideally, laws ratify already developed  
 social consensus.” He called for ﬁrms to develop business models that would  
 work with the grain of the information age. His band, the Grateful Dead, let  
 people tape their songs from the 1970s, and became one of the biggest stadium  
 draws. He suggested that other industries explore models of live performance  
 and service rather than selling bundles of bits [170].

There was vigorous debate and innovation on the copyright front during the

dotcom boom of the later 1990s. Quite apart from arguments about books, jour-  
 nals, music and ﬁlms – to which we will return shortly – there was a growing  
 realisation of the need for shared infrastructure and tools. Many common com-  
 ponents of the communications infrastructure, such as BGP, DNS and SMTP,  
 had been ﬁrst implemented at taxpayer expense and published, and ﬁrms often  
 found they needed to add still more code to the commons. For example, after  
 Netscape made available the ﬁrst popular web browser in 1994, Microsoft killed  
 them by giving away its own browser, Internet Explorer, free with Windows, and  
 tried to create a monopoly at the server side with a product then called Internet  
 Information Server which it launched in 1995. Other ﬁrms who were racing to  
 establish a presence in the growing e-commerce industry were so alarmed at the  
 prospect of Microsoft extracting all the value that they set up Apache, which  
 became the leading web server the following year. This may have been one of  
 the most important pieces of software ever written, as it meant that Microsoft  
 could not control both ends of the link in the early days of the web, so they could  
 not turn it into something proprietary from which they could extract rent. As a  
 result, the web remained open for many years, and it was possible for companies  
 such as Google and Facebook to get going. (We may now have a policy struggle  
 with them instead, but a lot of innovation happened meantime.)

Moving from the policy to the mechanics, when software engineers – or book

authors or musicians – place works in the public domain, we have a wide range  
 of conditions we may want to attach. Some writers are happy for their work  
 to be used by anyone, so opt for a BSD-style license; others want their work to

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 746 | Ross Anderson |

*24.2. COPYRIGHT*

remain in the commons rather than being incorporated into closed proprietary  
 products, so prefer the GPL; academics generally want our stuff to be used  
 provided we’re acknowledged as the creators. In 2001 Larry Lessig founded the  
 *Creative Commons* (CC) to bring some order to this; it makes available a set of  
 licenses which parametrise this and enable you to specify how your work may  
 be used. For example, you can specify whether a user can share your work with  
 others; whether commercial uses are allowed; whether they must give you proper  
 attribution; whether they can adapt and build on it, and if so whether they have  
 to distribute their contributions under the same license as the original. These  
 licenses are now used widely outside of software. In fact, most of my academic  
 papers are available under CC licenses, and my agreement with the publishers  
 of this book speciﬁes that I may make all the chapters available freely online  
 42 months after the manuscript is sent for publication. I appreciate it if you  
 pay for the book, but I want it to be available to everybody – even if the latest  
 versions go online after a delay.

A critical development came in 1996 with section 230 of the US Communi-

cations Decency Act (CDA). This let the online service providers off the hook  
 on copyright law by stating that ‘No provider or user of an interactive computer  
 service shall be treated as the publisher or speaker of any information provided  
 by another information content provider’ – making ﬁrms like Google and Face-  
 book possible, and leaving the corporate lawyers to chase individual ﬁle sharers.  
 The service ﬁrms are supposed to take down infringing content when they’re no-  
 tiﬁed of it; in practice, the boundaries are hard to police, and the incentives are  
 perverse (section 230 shelters them when they run ads for counterfeiters [1830]).  
 We’ll return to this later, in this chapter and in Chapter 26.

So there are many alternative business models, both for software and for

other products of human creativity. One is *freemium*: you give away a basic  
 version of the product, and sell a premium version. (Even once this book is free  
 online as PDF ﬁles, you’ll have to pay money for a printed book.) Another is to  
 give your software away free, and make your money from selling services, from  
 advertising, or by acting as spyware and selling data about the user. You can  
 combine them: get customers addicted to your free product, and then sell them  
 more storage or an ad-free experience. The success of these models in software –  
 with the Linux industry living from consulting and Google from ads – suggested  
 a similar approach to other online businesses.

In the second edition of this book in 2008, I suggested then that“the solution

for Hollywood’s problem lies in a change of business model.” As this third edition  
 went to press in August 2020, the New York Times was lamenting the death of  
 Hollywood [1791]. The studio that led Hollywood, Warner, had its executives  
 ﬁred, without the usual golden parachutes; no longer masters of the universe,  
 they had become the employees of the video production arm of a phone company.  
 The ﬁlm industry had changed from a wholesale business which did deals with  
 distributors over a handshake by the pool into a retail one where maximising  
 subscription revenue is the core skill. The only studio to remain in recognisable  
 form is Disney, which managed the transition to subscription early – helped  
 perhaps by having Steve Jobs as its largest shareholder and as a main board  
 director.

I will return to copyright policy later in section 24.5, but let’s now take a

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 747 | Ross Anderson |

*24.2. COPYRIGHT*

quick historical tour at the world of protecting media content.

**24.2.3** **Books and music**

In 1800, there were only 80,000 frequent readers in England; most of the books  
 up till then were serious philosophical or theological tomes. After the invention  
 of the novel, a mass market appeared for books, and circulating libraries sprang  
 up to service it. The educated classes were appalled, and printers were frightened  
 that the libraries would deprive them of sales. But the libraries so whetted

people’s appetite for books that the number of readers grew to 5,000,000 by  
 1850. Sales of books soared as people bought books they’d ﬁrst borrowed from  
 a library. The library movement turned out to have been the printers’ greatest  
 ally and helped create a whole new market for mass-market books [1718].

People have been copying music much longer than software. Paganini was

so worried that people would copy his violin concertos that he distributed the  
 scores himself to the orchestra just before rehearsals and performances, and  
 collected them again afterwards. (As a result, many of his works were lost to  
 posterity.)

Copyright *collecting societies* were established from the mid-19th century,

starting in Paris; composers who were members would charge venues or bands  
 a fee for performing their compositions. In many countries these have become  
 monopolies backed by law; to perform at our university’s concert hall, you have  
 to pay the Performing Rights Society a levy. You can submit them a playlist,  
 and if you play all your own compositions then some of the money may ﬁnd  
 its way back to you eventually. Many tunes are *orphan works* in that their

composers’ heirs are unknown, so the societies can either keep the money or  
 share it among their known composers. The free culture movement and the  
 pirate parties advocate restricting or abolishing copyright in order to erase such  
 injustices; but while they’ve won a few parliamentary seats in some European  
 countries, they always seem to be outgunned by the copyright lobbyists on the  
 world stage (an issue to which I’ll return later in section 24.5.1).

When the cassette recorder came along in the 1960s, the record industry

lobbied for (and in some countries got) a tax on audiocassettes, to be distributed  
 to copyright holders. Technical measures were also tried. The Beatles’ record  
 Sergeant Pepper contained a 20KHz spoiler tone that should in theory have  
 combined with the 21KHz bias frequency of the tape to produce a 1KHz whistle  
 that would spoil the sound. In practice it didn’t work, as many record players  
 didn’t have the bandwidth to pick up the spoiler tone. But in practice this  
 didn’t matter. Cassettes turned out not to be a huge problem because sound  
 quality is noticeably poorer on home equipment; people mostly used them to  
 record music to listen to in their cars. Then, in the 1980s, the arrival of the  
 Sony Walkman made cassettes into big business, and although there was some  
 copying, there were huge sales of pre-recorded cassettes and the music industry  
 cleaned up.

Audio copying became a headline concern again in the 1990s, thanks to the

MP3 format for compressing audio. Previously, digital audio was protected by  
 its size: a CD of uncompressed music can take 650Mb. However, MP3 enables

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 748 | Ross Anderson |

*24.2. COPYRIGHT*

people to squeeze an audio track into a few megabytes, and broadband enables  
 ﬁles of this size to be shared easily. By 1998, some 40% of the network traffic  
 at MIT was MP3 traffic.

The industry response was to push for technical ﬁxes. This led to the growth

of the rights-management industry. It had its origins in work on digital pub-  
 lishing and in the mechanisms used to protect pay-TV and DVDs, so let’s take  
 a quick look at those ﬁrst.

**24.2.4** **Video and pay-TV**

The early history of videocassettes was a replay of the history of audio cas-  
 settes. At ﬁrst Hollywood was terriﬁed, and refused to release movies for home  
 viewing. Crude technical measures were taken to prevent copying – such as  
 the Macrovision system which added spurious synchronization pulses to confuse  
 the recording circuitry of domestic VCRs – which again turned out to be easy  
 to defeat. Then Hollywood became paranoid about video rental stores, just as  
 book publishers had been about libraries. Once more, libraries turned out to  
 be the publisher’s friend, as being able to rent videos got people to buy VCRs  
 and whetted their desire to own their favorite movies. VCRs and videocassettes  
 became mass-market products rather than rock stars’ toys, and by 2000 sales  
 of prerecorded cassettes made up most of the income of ﬁrms like Disney. The  
 business model changed so that the cinema release was really just advertising  
 for the sales of the video.

By then, many of the world’s pre-teens demanded that their parents build

them a collection of Disney cassettes, just like their friends had, so a videocas-  
 sette pirate had to make the packaging look original. This reduced the problem  
 to an industrial counterfeiting one. As with mass-market software before the  
 onset of online registration, or with perfumes and Swiss watches today, enforce-  
 ment involves sending out ﬁeld agents to buy products, look for forgeries, trace  
 the supply chain and bring prosecutions.

More interesting technical protection mechanisms were built into broadcast

pay-TV equipment.

The advent of pay-TV, whether delivered by cable or satellite, created a

need for *conditional access* mechanisms which would allow a station operator to  
 restrict reception of a channel in various ways. If the operator had only bought  
 the rights to screen a movie in Poland, they’d have to block German or Russian  
 viewers within the satellite footprint from watching. Porn channel operators  
 needed to prevent reception in countries like Ireland with strict censorship laws.  
 Most operators also wanted to be able to charge extra for speciﬁc events such  
 as boxing matches.

**24.2.4.1** **Typical system architecture**

The evolution of early systems was determined largely by the hardware cost of  
 deciphering video (for a history of set-top boxes, see [425]). The ﬁrst generation  
 of systems, available since the 1970s, were crude analog devices which used  
 tricks such as inverting the video signal from time to time, interfering with

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 749 | Ross Anderson |

*24.2. COPYRIGHT*

the synchronization, and inserting spikes to confuse the TV’s automatic gain  
 control. They were easy enough to implement, but also easy to defeat; breaking  
 them didn’t involve cryptanalysis, just an oscilloscope and persistence.

The second generation of systems appeared in the late 1980s and employed

a hybrid of analog and digital technologies: the broadcast was analog, but the  
 subscriber control was digital. These included systems such as Videocrypt and  
 Nagravision, and typically had three components:

*•* a subscription management service at the station enciphers the outgoing  
 *entitlement control messages* (ECMs) in it, and issues access tokens such  
 as smartcards to subscribers;

*•* a *set-top box* converts the cable or satellite signal into one the TV can deal

*•* the subscriber smartcard personalises the device and controls what pro-  
 preting the ECMs and providing keys to the descrambling circuit in the  
 set-top box.

This architecture means that the complex, expensive processes such as bulk

video scrambling could be done in a mass-produced custom chip with a long  
 product life, while key-management functions that may need to be changed  
 after a hack can be sold to the customer in a low-cost token that is easy to  
 replace. If the set-top box itself had to be replaced every time the system was  
 hacked, the economics would be much less attractive4.

The basic mechanism is that the set-top box decodes the ECMs from the

input datastream and passes them to the card. The card deciphers the ECMs to  
 get both control messages (such as “smartcard number 123356, your subscriber  
 hasn’t paid, stop working until further notice”) and keys, known as *control*  
 *words*, that are passed to the set-top box. The set-top box then uses the control  
 words to descramble the video and audio streams. There’s a detailed description  
 in [456].

**24.2.4.2** **Video scrambling techniques**

Because of the limitations on the chips available at low cost in the early 1990s,  
 hybrid systems typically scrambled video by applying a transposition cipher  
 to picture elements. A typical scheme was the *cut-and-rotate* algorithm used  
 in Videocrypt. This scrambles one line of video at a time by cutting it at

a point determined by a control byte and swapping the left and right halves  
 (Figure 24.1):

This involved analog-to-digital conversion of the video signal, storage in a

buffer, and digital-to-analog conversion after rotation – a process which could  
 just about be shoehorned into a low-cost custom VLSI chip by 1990. However,

4Now that set-top boxes cost a few dollars, and the shipping costs dominate, the smartcard

is often just soldered to the motherboard and the whole box is replaced if there’s a hack.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 750 | Ross Anderson |

*24.2. COPYRIGHT*

|  |  |  |  |
| --- | --- | --- | --- |
| Plain� | t� | Cipher� | t� |

Figure 24.1: – cut-and-rotate scrambling

a systemic vulnerability of such systems is that video is highly redundant, so  
 it may be possible to reconstruct the image using ‘oscilloscope and persistence’  
 techniques, enhanced by simple signal processing. This was ﬁrst done by Markus  
 Kuhn in 1995 and required the use of a university supercomputer to do in real  
 time. Figure 24.2 shows a frame of enciphered video, and Figure 24.3 the same  
 frame after processing. By 2000, it was possible to do this on a PC [1824]. If  
 this attack had been feasible earlier, it would have given a complete break of the  
 system, as regardless of how well the smartcard managed the keys, the video  
 signal could be retrieved without them. Hybrid systems are still used by some  
 stations in less developed countries, together with frequent key changes to make  
 life inconvenient for the pirates – whose problem is to distribute the keys to  
 their customers as they crack them.

The major developed-world operators moved to digital systems in the early

2000s. These digital systems work on the same principle – a set-top box with  
 the crypto hardware and a smartcard to hold the personal keys that in turn  
 decipher the content keys from ECMs. However the crypto now typically uses a  
 block cipher to protect the entire digital video stream. I’ll describe the current  
 digital video broadcast systems in the next section.

The hybrid scrambling techniques lasted (just) long enough. However, they

have some interesting lessons to teach, as they were subjected to quite deter-  
 mined attack in the decade after 1995, so I’ll go brieﬂy through what went  
 wrong.

|  |  |
| --- | --- |
|  |  |

**Figure 24.2 – scrambled video frameFigure 24.3 – processed video frame**

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 751 | Ross Anderson |

*24.2. COPYRIGHT*

**24.2.4.3** **Attacks on hybrid scrambling systems**

Given a population of set-top boxes that can use a stream of control words to  
 unscramble broadcast video, the next problem was to ensure that only paying  
 customers could get the control words. In general, this could be done with

allow and deny messages. But the bandwidth available was typically of the

order of ten ECMs per second. So sending an allow message to each of ﬁve  
 million subscribers would take over a week, and deny messages were mostly  
 used instead.

The customer smartcard interprets the ECMs. If the current programme is

one the subscriber is allowed to watch, then a keyed hash – essentially a message  
 authentication code (MAC) – is computed on a series of ECMs using a master  
 key held in the card and supplied to the set-top box as the control word:

*CW* = *MAC*(*K*; *ECM*1*, ECM*2*, ECM*3*, ECM*4)

So if a subscriber stops paying their subscription, their card can be inacti-

vated by sending an ECM ordering it to stop issuing control words; and it needs  
 access to the ECM stream in order to compute control words at all. Provided  
 the cards can be made tamper-resistant, only compliant devices should have  
 access to the master key *K*, and they should commit suicide on demand. So  
 what could go wrong?

The ﬁrst attacks were on the protocol. Since the control word sent from the

smartcard is the same for every set-top box currently unscrambling the program,  
 one person can record the stream of control words, by placing a PC between the  
 smartcard and the set-top box, and post them online. Other people can video-  
 record the scrambled program, and unscramble it later [1255]. Servers sprung  
 up for this *key-log attack*, but were only a minor nuisance to the industry; not  
 many viewers were prepared to buy or build a special adapter to connect their  
 PC to their set-top box. Hobbyists with such equipment found other attacks  
 including *blockers*, programs that would prevent ECMs addressed to your card  
 from being delivered to it; this way, you could cancel your subscription without  
 the operator being able to cancel your service [1255].

Cryptanalysis also gave some opportunities. Every half-second or so the

smartcard supplies the set-top box with a new control word, and this is loaded  
 into a keystream generator which works as follows. There are two linear feedback  
 shift registers, of lengths 31 and 29 in the Eurocrypt system, which generate  
 long linear sequences. Some of the bits of register 1 are used as address lines to  
 a multiplexer, which selects a bit from register 2; this bit becomes the next bit  
 of the keystream sequence. Each successive byte of output becomes a control  
 byte for the scrambler (Figure 24.4).

|  |
| --- |
| Linear feedback shift register 1 |

|  |  |  |
| --- | --- | --- |
| *#* *#* *#*  (address)   |  | | --- | | Multiplexer |   (select) | *�!* output |

*"* *"* *"* *"* *"*

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 752 | Ross Anderson |

*24.2. COPYRIGHT*

|  |
| --- |
| Linear feedback shift register 2 |

*Figure 24.4 – the multiplexer generator*

|  |
| --- |
| The designers intended that breaking this cipher should involve guessing the  key, and as this is 60 bits long a guess would take on average 259 trials which is |
| uneconomic – as it has to be done about twice a second. But it turns out that  the cipher has a shortcut attack. The trick is to guess the contents of register 1,  use this address information to place bits of the observed keystream in register  2, and if this causes a clash, reject the current guess for register 1. (I discovered  this attack in 1985 and it’s what got me interested in cryptography.) The high-  order four bits or so of each control word are easy to deduce from inter-line  correlations – it’s the least signiﬁcant bits you really have to work hard for. So  you can reconstruct the latter using cryptanalysis. But this computation is still  of interest to hobbyists rather than the mass market. |

Perhaps the most powerful of the ‘amateur’ attacks exploited a master-key

leakage: someone who bought a second-hand PC, looked at the hard disk out  
 of curiosity, and managed to undelete a complete subscriber management sys-  
 tem for one pay-TV operator, including embedded master keys. This enabled  
 enthusiasts to write software to emulate a subscriber smartcard completely – in  
 fact, it could even be ‘improved’ so it would not turn itself off when ordered to  
 do so by an ECM.

Anyway, the commercial pirates turned to reverse engineering smartcards

using microprobing techniques, and in section 18.5 I described the arms race  
 that followed. But hardware ﬁxes were limited to new card issues, and the

operators didn’t want to issue a new card more than once a year as it cost  
 several dollars per subscriber, and the subscriptions were usually less than $20  
 a month. So other defensive techniques were tried too.

Litigation was one route, but it took time. A lawsuit was lost against a pirate

in Ireland, which for a while became a haven from which pirates sold cards by  
 mail order all over Europe. The industry’s lobbying muscle was deployed to  
 bring in European law to override Dublin, but this took years. By the middle of  
 1995, the main UK satellite TV station (Sky-TV) was losing 5% of its revenue  
 to pirate cards, mostly sold by mail order from Dublin.

So all through the mid 1990s, pirates and the operators engaged in a war of

technical countermeasures and counter-countermeasures. The operators would  
 ship a new card, and within months the pirates would have reversed it and  
 be offering clones for sale. The operators would buy some, analyze them, and  
 develop tricks to cause them to fail. The problem faced by the operators was  
 this: when all the secrets in your system can be compromised within months,  
 how can you still ﬁght back against the pirates without having to reissue all the  
 cards?

The operators came up with all sorts of cunning tricks. One of their more

effective ones was an ECM whose packet contents were executed as code by  
 the smartcard; in this way, the existing card base could be upgraded on the  
 ﬂy and implementation differences between the genuine and pirate cards could  
 be exploited. Any computation that would give a different answer on the two  
 platforms – even if only as a result of an unintentional timing condition – could

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 753 | Ross Anderson |

*24.2. COPYRIGHT*

KM

|  |  |
| --- | --- |
| KG11 | KG12 |

KG22

|  |  |
| --- | --- |
| KG21 | • • •� |

• • •ffi

• • •ffi

KGn1

|  |  |
| --- | --- |
| k1 | • • •� |

k2

Figure 24.5: – binary revocation tree

be fed into the MAC algorithm to make the pirate cards deliver invalid control  
 words.

One of the systems (Eurocrypt) had an efficient revocation scheme designed

in from the start, and it’s worth looking at brieﬂy. Each of the subscriber smart-  
 cards contains a subscriber key *ki*, and there is a binary tree of intermediate  
 group keys *KGij* linking the subscriber keys to the currently active master key  
 *KM*. Each operational card knows all the group keys in the path between it  
 and the master key, as in Figure 24.5.

In this scheme, if (say) key *k*2 appears in pirate cards and has to be revoked,

|  |
| --- |
| then the operator will send out a stream of packets that let all the other sub- |
| scriber cards compute a new master key *KM*. The ﬁrst packet will be *{K0M}KG*12  *M* at once; then there will be a  *K0*  group key *KG0*11 encrypted under *GK*22; and so on. The e↵ect is that even*M* encrypted under an updated version of *KG*11: *{K0M}KG0*11; then this new |
| with ten million customers the operator has to transmit less than ﬁfty ECMs  in order to do a complete key change. Of course, this isn’t a complete solution:  one also needs to think about how to deal with pirate cards that contain several  subscriber keys, and how leaked keys can by identiﬁed without having to go to  the trouble of breaking into pirate cards. But it’s a useful tool in the box. |

Psychological measures were also used. For example, one cable TV station

broadcast a special offer for a free T-shirt, and stopped legitimate viewers from  
 seeing the 0800 number to call; this got them a list of the pirates’ customers.  
 Economic factors also matter here, as everywhere. Pay-TV pirates depend for  
 their success on time-to-market as much as conventional software ﬁrms: a pirate  
 who could produce a 99% correct forgery in three weeks would wipe out a  
 competitor who produced a 99.9% forgery after three months. So pirates race

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 754 | Ross Anderson |

*24.2. COPYRIGHT*

to market just like legitimate vendors, and pirate cards have bugs too. An

understanding of economics teaches that it’s best to let a pirate build up a  
 substantial user base before you pull the plug on him, as this gives him time to  
 wipe out his competitors, and also as switching off his cards once he’s established  
 will destroy his credibility with more potential customers than an immediate  
 response would. But if you leave him too long, he may acquire the ﬁnancial and  
 technical resources to become a persistent problem.

The pay-TV industry learned to plan in advance for security recovery, and to

hide features in their products that weren’t used initially but could be activated  
 later5.

Eventually, the smartcards were made much harder to forge by including

proprietary encryption algorithms in the processor hardware. As the attacker  
 couldn’t simply read out the algorithm with a probing station but had to reverse  
 engineer thousands of gates in the chip, they reduced to a handful the number  
 of laboratories with the technical capability to do attacks. Many of these labo-  
 ratories were drawn into the industry’s orbit by consultancy deals or other kinds  
 of sponsorship. Those who remained outside the tent were watched. Vigorous  
 legal enforcement provided the last link in the chain. The industry hunted down  
 the main commercial pirates and put them out of business, whether by having  
 them jailed or by drowning them in litigation.

In the last big pay-TV piracy case in the 20th century, British pirate Chris

Cary was convicted of forging Sky-TV smartcards whose design he had had  
 reverse engineered by a company in Canada for $105,000. He sold forgeries

through a front company in Ireland, where counterfeit cards were not illegal  
 yet [1368]. So Sky TV’s security consultants inﬁltrated a spy into his Dublin  
 sales office, and she quietly photocopied enough documents to prove that the  
 operation was really being run from the UK [956]. The British authorities didn’t  
 want to prosecute, so Sky brought a private prosecution and had him convicted.  
 When the authorities put him in an open prison and he absconded, Sky’s private  
 detectives relentlessly hunted him down and caught him in New Zealand, where  
 he’d ﬂed using a passport in a dead person’s name [847]. He then ended up in  
 a proper jail. Sky-TV’s relentless unpleasantness served as a warning to others.

**24.2.4.4** **DVB**

Digital video broadcasting (DVB) largely operates using a set of standards that  
 have evolved over the years since 1996 and that are controlled by the DVB  
 Consortium, an industry group of over 250 members. The standards are many  
 and complex, relating to IPTV and digital terrestrial TV as well as satellite  
 TV, and to free-to-air services as well as pay-TV. DVB has been replacing  
 analog/hybrid systems, starting with the UK and Germany in 2003. The latest  
 standards, DVB-T2, were promulgated by ETSI in 2009.

The protection mechanisms are complex, and some of them are covered by

nondisclosure agreements, but here is a telegraphic summary. The conditional  
 access mechanisms for DVB are similar to the hybrid system: the content en-

5We discussed in section 16.3.1 how banknote printers learned years ago to include a whole

series of security printing features that could be disclosed one at a time as needed.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 755 | Ross Anderson |

*24.2. COPYRIGHT*

cryption is digital, but the keys are generated by subscriber smartcards oper-  
 ating on EMMs and ECMs as before. The encryption uses the DVB Common  
 Scrambling Algorithm, which was available only under NDA, but leaked in 2002.  
 In 2011, an attack was found by Erik Tews, Julian W¨alde and Michael Weiner,  
 which was then barely practical as it requires an 8TB rainbow table [1872]. The  
 smartcards are not standardised (except at the interface level) so each broad-  
 caster can use his favorite crypto tricks and suppliers; the piracy to date seems  
 to have involved smartcard cloning, and there have been various lawsuits where  
 pay-TV operators have accused each other of hacking.

Pay-TV, whether cable or satellite, peaked in 2008 with 75% of US house-

holds. What dislodged it was Netﬂix, and more generally the move to online  
 subscription services based on broadband.

**24.2.5** **DVD**

The history of DVD was both a warning of trouble to come between Holly-  
 wood and the computer industry, and an object lesson on how not to do copy  
 protection.

The consumer electronics industry introduced the *digital video disk* (DVD),

later renamed the *digital versatile disk*, in 1996. As usual, Hollywood took fright  
 and said that unless DVD had a decent copy protection mechanism, ﬁrst-class  
 movies wouldn’t be released for it. So a mechanism called the *content scrambling*  
 *system* (CSS) was built in at the last minute; arguments over this held up the  
 launch of DVD and it was designed in a rush. (The story of how the DVD  
 standards evolved is told in Jim Taylor’s standard reference [1865], which also  
 describes most of them.)

DVD had *region coding*: disks were supposed to run only on players from

some designated list of regions, to support the traditional practice of releasing  
 a movie in the USA ﬁrst, then in Europe and so on, in order to minimise the  
 cost of producing physical ﬁlm prints, and the ﬁnancial loss if the ﬁlm bombs.  
 But users preferred to buy DVD players in which region coding could be turned  
 off. So every DVD vendor wanted to have the second most insecure player on  
 the market; they didn’t want to be the ﬁrm that Hollywood was beating up on,  
 but they wanted prospective customers to be conﬁdent that their player’s region  
 coding could be hacked.

This left CSS, which was known to be vulnerable by the time that DVD

was launched [1494]. It has a keylength of 40 bits so the equipment wouldn’t  
 fall foul of US export regulations, but the design was so poor that the effective  
 keylength was only 16 bits. A Norwegian teenager, Jon Lech Johansen, reverse  
 engineered the algorithm and wrote decryption software for it, DeCSS. Industry  
 lawyers got injunctions against people who put it online but these were seen as  
 censorship, so it started appearing on websites outside the USA, on T-shirts,  
 in songs, and in other forms of speech that traditionally enjoy constitutional  
 protection6. This just got it distributed ever more widely, and made Hollywood  
 look foolish [1127]. Their lawyers blundered on, persuading the government of

6There was a full description of CSS and how to break it in the ﬁrst and second editions

of this book; as DVDs are going the way of the dinosaur, I’ve dropped it for this edition.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 756 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

Norway to prosecute Johansen. He was acquitted on appeal in 2003.

Another set of problems came from the fact that the PC is an open plat-

form. The DVD consortium required people producing DVD player software to  
 obfuscate their code so that it would be hard to reverse engineer. Papers duly  
 appeared on tricks for systematic software obfuscation [141]. But this closed  
 approach came into conﬂict with Linux, the open-source PC operating system  
 that was already used by millions of people. The DVD consortium’s philosophy  
 was not consistent with making DVD drivers available to the Linux community.  
 So as PCs with CD drives started being replaced in the shops with PCs ﬁtted  
 with DVD drives, the Linux user community either had to break CSS, or give  
 up using Linux in favour of Windows. Under the circumstances, it was only a  
 matter of time before someone ﬁgured out CSS and DeCSS appeared.

Anyway, DVD followed the usual pattern: Hollywood terriﬁed, and refusing

to release their best movies; technical measures taken to prevent copying, which  
 quickly got broken; then litigation. I wrote in 2001: “A reasonable person might  
 hope that once again the studios will see sense in the end, and make a lot of  
 money from selling DVDs. There will be copying, of course, but it’s not entirely  
 trivial yet – even a DSL modem takes hours to send a 4Gb DVD movie to a  
 friend, and PC disk space is also an issue.” This came true; although some  
 studios held out for a year or two, they all climbed on the DVD bandwagon,  
 and by the second edition in 2008, Disney was making most of its money from  
 DVD sales.

There was then an attempt to market higher-density optical media, with a

format war in 2007 between HD-DVD and Blu-Ray, which both used shorter  
 wavelength lasers to encode information more densely giving up to 50Gb per  
 disk. Both used the Advanced Access Content System (AACS), which I de-  
 scribed in the second edition of this book. However, only the Playstation 3 did  
 a full implementation of Blu-Ray, and HD-DVD never got real traction at all.  
 They were destroyed, as distribution media, by the growth of broadband, and  
 as storage media by the falling cost of USB memory sticks.

**24.3** **DRM on general-purpose computers**

Victor Shear patented self-destruct software in the 1980s and his company be-  
 came InterTrust [1793]; their DigiBox system is described by Olin Sibert, David  
 Bernstein and David Van Wie in [1735]. This enabled a DRM mechanism to  
 reﬂect real-world ownership, so that I could sell you a photo and you’d be able to  
 decrypt it once you had the receipt; what’s more, you could give it to somebody  
 else after which you’d no longer have it.

Intertrust were the most successful of a number of ﬁrms who worked in

the mid-90s on ways to control the sale and distribution of digital goods over  
 the Internet to customers with personal computers7. The original applications  
 included the distribution of newspapers and articles from scientiﬁc journals [315],

7The InterTrust patents were one of only four computer-related patents from the 20th

century that caused a nine-ﬁgure sum to change hands, the others being the Harvard virtual  
 memory patents, the RSA public-key patents and the Fraunhofer MP3 patents.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 757 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

although it was always understood that music and video would follow once  
 networks had enough bandwidth.

The basic problem is that a PC, being a general-purpose computer, can in

principle copy any ﬁle and send it to any other computer; unlike with analog  
 copying, copies are perfect, so millions of copies might be made from one original.  
 The problem is compounded by the fact that, from the viewpoint of the content  
 vendor, the PC owner is the ‘enemy’. The music industry believed that unlimited  
 copying would destroy their business; the computer industry told them that  
 DRM was intrinsically impossible on a general-purpose computer, so they’d  
 better get a new business model. The music and ﬁlm industries, despite being  
 a tenth of the computer industry’s size, had much more clout in Congress (a  
 Microsoft guy complained that the average Congressman was much keener to be  
 photographed with Madonna than with Bill), and they still controlled access to  
 the music and video that the computer industry wanted their PCs and phones  
 to be able to play. The result was a push for DRM.

**24.3.1** **Windows Media Rights Management**

Windows Media Player (WMP) was an early deployment of DRM, replacing an  
 earlier media player when Windows 98 was released. It enabled a user to play  
 music, watch video and view photos, with features ranging from MP3 player  
 support to synchronisation of lyrics for karaoke. It introduced *Windows Media*  
 *Rights Management* (WMRM), which works as follows.

A store wanting to sell digital media encrypts each item using a content key

and puts the encrypted ﬁles on a streaming media server linked to their web site.  
 In order to access a media object, the customer must get hold of a license, which  
 consists of the object identiﬁer, the license key seed, and a set of instructions in  
 a *rights management language* which state what they can do with it; how many  
 times they may play it, whether they can burn it to a CD, and so on. The license  
 is generated by a license server and encrypted using a public key generated by  
 the customer’s WMP application. License acquisition may involve registration  
 or payment, but it may also happen silently in the background [1558].

The architecture is similar to pay-TV conditional access, in that the bulk

encryption task of protecting the music or video is separated from the person-  
 alised task of key management, so the video doesn’t have to be encrypted anew  
 for each customer. And just as pay-TV smartcards can be replaced when keys  
 are leaked or the key management mechanism compromised, so the key manage-  
 ment functions of WMRM are performed in an ‘individualized blackbox’ (IBX)  
 component of the software, which gets replaced as needed during the Windows  
 update process.

The IBX internals have been reverse-engineered from time to time [1693].

The customer’s private key is obscured by the blackbox and hidden in a ﬁle;  
 licenses the customer has previously acquired are kept in a license store; con-  
 tent keys are encrypted using the customer’s public key; and the protocol gets  
 tweaked from time to time as Microsoft has to recover from hacks. I described in  
 section 6.2.5 how in the early 2000s Microsoft, Intel and some other big players  
 formed the Trusted Computing Group to try to build DRM properly into the

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 758 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

PC architecture. The attempt failed for both business and technical reasons,  
 but led to TPM chips for trusted boot, to TrustZone enclaves in Arm processors,  
 and eventually to SGX enclaves in Intel chips.

Microsoft launched *Information Rights Management* (IRM) with Windows

Server 2003, which aimed to extend DRM to general users; the idea was that  
 access controls over a document or other digital object would be retained by  
 its creator. So DRM wouldn’t just beneﬁt Hollywood; I could send you an

email that you could only read, and never copy, and that would vanish after a  
 month. The vision was that this would be supported by Trusted Computing  
 mechanisms across the entire Windows ecosystem, and conveniently fortify the  
 ecosystem against challenges from the likes of Linux or Google docs. Corporate  
 America didn’t like the lock-in, though, and Microsoft couldn’t get the operating  
 system mechanisms to work. Nowadays, it’s easy to implement such distributed  
 use controls in cloud-based systems such as Office365 or Gmail, but it’s too hard  
 to work across such ecosystems; so we’ve ended up not too far from where we  
 might have been had Trusted Computing been made to work.

WMRM was then replaced in Windows 10 by PlayReady, a newer Microsoft

‘media ﬁle copy prevention technology’. WMP is used at its most basic to pro-  
 vide a streaming media service, to support music subscription services, and  
 geographically-linked services, such as MLB.com which makes major league  
 baseball games available everywhere except in the team’s home area – for which  
 the rights have usually been sold to local TV stations.

**24.3.2** **Fairplay, HTML5 and other DRM systems**

The Microsoft offering was fairly typical of rights-management systems. Apple’s  
 FairPlay, which was launched in the iPod and in its media player QuickTime,  
 also has tunes encrypted under master keys. When a tune is bought the cus-  
 tomer is sent the master key encrypted under a random session key, plus the  
 session key encrypted under his iTunes player’s RSA public key. Session keys are  
 backed up online on Apple’s servers. As with Windows, a number of programs  
 have appeared from time to time that unlocked protected content, and Apple  
 duly upgraded iTunes. Apple iTunes was replaced with Apple Music in 2020.

Some ﬁrms’ rights-management systems were downright abusive, and a par-

ticularly extreme case arose in 2005 with Sony’s XCP system. The ﬁrst time  
 a user inserted a CD with this system into a PC, it presented an end-user li-  
 cense agreement; if the user declined, the CD was ejected, and if they accepted  
 it loaded and hid a rootkit that intercepted all accesses to the CD drive and  
 prevented Sony music being played by any other media player. Microsoft classi-  
 ﬁed it as malware and had it removed by Windows Defender and the Malicious  
 Software Removal Tool [1307]. It later turned out that Sony had even included  
 in their rootkit some software that violated the copyrights of others.

There was signiﬁcant controversy in 2012–14 when the World Wide Web

Consortium (W3C) was debating whether to adopt HTML5 which provides for  
 a sandbox in browsers to support multimedia content with DRM, and Encrypted  
 Media Extensions (EME) as a means for the software in the sandbox to com-  
 municate with online license managers. When they eventually went ahead in

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 759 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

2014, W3C chair Tim Berners-Lee was ﬁercely criticised for adopting a stan-  
 dard which excludes open-source browsers in the future. Since 2017, browsers  
 need to license ‘Widevine’ DRM software from Google to support services such  
 as Netﬂix. Mozilla was the last major browser to switch, after they concluded  
 that refusing would just cause most of their users to switch browsers. In 2020,  
 Google stopped supplying this technology to open-source browsers; thereafter  
 all new browsers will have to be proprietary; this had been predicted by EFF  
 during the debate in 2012–4 [571].

The other development in 2020 is Microsoft’s launch of “double encryption”,

a kind of DRM to make regulated industries like banking happier about keeping  
 sensitive data in the Office365 / Azure cloud: content keys are kept on the  
 local device, but the whole thing is integrated with the Microsoft structure of  
 access controls [432]. Whether DRM operated by Microsoft would stop an FBI  
 agent armed with a FISA warrant getting access to data on a Microsoft cloud  
 is an interesting question; I suppose we’ll only know the answer when the next  
 Snowden comes out.

**24.3.3** **Software obfuscation**

As I already mentioned, early software protection mechanisms used software  
 obscurity to hide keys and to check for the presence of machine ﬁngerprints,  
 dongles and license servers. Kids with disasssemblers and time on their hands  
 tended to defeat such tricks, so where possible ﬁrms would move some critical  
 functionality to the cloud, to trustworthy hardware, or both.

But that is not always possible, and in 2020 the critical applications include

*runtime application self-protection* (RASP). As I discussed in section 12.7.4, this  
 is a set of techniques used by some mobile app developers to protect apps on  
 phones that may have been rooted or jailbroken by malware. It’s used by Face-  
 book to protect customers using its Android app in less developed countries  
 where many Android phones are secondhand, out of patch support and rooted,  
 often by local sales agents. And following a mandate from the European Cen-  
 tral Bank, RASP is becoming mandatory for banking apps in Europe, or for  
 authenticator apps on which they rely. In both cases the objective is to protect  
 cryptographic keys from an attacker who roots the device. This was also the  
 threat model for 1990s products such as Windows Media Player.

There were early attempts to write obfuscating compilers that would produce

tamper-resistant software; an early Intel project is described at [141] and led to  
 tools used in early software DVD players. These were duly broken, as I described  
 in section 24.2.5 earlier, and led Intel to move towards Trusted Computing and  
 eventually SGX, as I described in section 6.3.1.

Theoretical computer scientists have written many papers on obfuscation

and indistinguishability; a seminal result by Boaz Barak and colleagues in 2001  
 suggests that we can’t write obfuscating compilers with strong and sustainable  
 protection properties [166]. But – as with other impossibility results in security  
 such as those on malware detection – the question then arises whether even if  
 perfect obfuscation isn’t possible in theory, practical obfuscation might be good  
 enough for some purposes.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 760 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

Microsoft moved to a philosophy of security renewal: the key-management

code for Windows Media Player was hidden in IBX and moved around, so it  
 might be in the Windows error handler one month and an obscure device driver  
 the next. Malware writers took a similar trajectory. As I described in sec-

tion 21.3.5, they often obfuscate their code by running it through a packer that  
 contains a polymorphic header which in turn decrypts the malware body. Keys  
 and headers are all different, making malware harder to recognise. Approaches  
 like this can sometimes be made to work moderately well, provided the main-  
 tainers are capable and motivated. Very often, though, they aren’t; naive ﬁrms  
 buying RASP from salesy vendors should expect the worst.

The main security research conferences have tended not to accept papers on

obfuscation as they see it as a tactical arms race rather than the accumulation of  
 scientiﬁc knowledge. There is nonetheless a small research community working  
 on obfuscation, and as of 2020 the state of the art when protecting an engine for  
 authentication or decryption is to implement a virtual machine that has an odd  
 instruction set, in which you implement the crypto, and then further obfuscate  
 the virtual machine itself (custom opcodes had already been used in Sky-TV  
 smartcards back in the 1990s). It is still a real problem though to evaluate such  
 a scheme, or even guess how much effort it will take to break it [555]. If a RASP  
 tester can’t extract the crypto key despite trying for a fortnight, that doesn’t  
 give you any guarantee against someone who tries for a month8. Decompilation  
 tools and techniques improve all the time, and many engineers spend much of  
 our lives trying to ﬁgure out what other people’s code actually does. Some

people acquire a real knack for this, but they might not be working in your  
 compliance testing lab! A lemons market is therefore to be expected.

All that said, there are some less heavyweight aspects to this. Some tools

obfuscate Java bytecode as they shrink and optimise it; one such, ProGuard,  
 is distributed as part of the Android SDK. And for entertainment, there’s the  
 International Obfuscated C Code Contest, where people have fun trying to hide  
 functionality in plain sight.

**24.3.4** **Gaming, cheating, and DRM**

Games were one of the ﬁrst applications of all – pretty well as soon as the  
 world’s ﬁrst proper computer, the EDSAC, was operational, research students  
 were writing games for it. Computer games have been big business for decades.  
 They drove the home-computer boom of the 1970s that in turn spawned the  
 PC industry; games consoles have been a huge market for microprocessors and  
 memory chips; and gaming – whether on consoles or PCs – has largely driven  
 the development of computer graphics [2056]. Game sales in the USA surpassed  
 movie box-office sales in 2001; and as games moved online, game ﬁrms started  
 to sell subscriptions, not just one-off tickets [280].

8I bear the scars personally. Back in the 1990s, Intel paid us to spend a fortnight trying

to hack a prototype DVD player binary that had been produced by Beelzebub, their internal  
 obfuscating compiler. We only got about halfway through, and the company then boasted to  
 its customers that ‘Cambridge couldn’t break this’. Jon Lech Johansen later spent a month  
 staring at the code and broke it, making us look stupid – but at least Intel ended up looking  
 stupider.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 761 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

When Nintendo moved console games into the home, they subsidised the

consoles from later sales of software cartridges and other add-ons, so a lot of  
 effort was put into controlling which accessories could be used, as I discuss later  
 in section 24.6; copy-protection of game software for PCs was also a big deal.  
 However the move to online computer games has mitigated these concerns. As a  
 critical part of the game logic runs on a server, the client software can be given  
 away, and the residual issue is whether players can get an unfair advantage.

There are very many ways in which gamers can cheat [2057]. Some games

ban collusion, such as contract bridge, and it’s hard to stop people playing on  
 an online platform from using an entirely separate channel to cheat. In the real  
 world, allegations of cheating are heard by a jury of experienced players, who  
 take a view on whether the outcome was better than could have been expected  
 in honest play. Even so, some decisions remain controversial for years: players  
 may be lucky, and partners who’ve played together for years may communicate  
 subconsciously without trying to. Online play can help as you can have online  
 records for statistical analysis, online tournaments where many players use the  
 same deal of cards, and new forms of play where people play with many partners  
 rather than just one.

Other games require collusion, such as adventure games involving teams of

people. As I discuss in section 8.6.8, these are currently, in 2020, the biggest  
 market for DDoS-for-hire services. Players, who are often schoolkids, pay a few  
 dollars for a service that will knock key members of the opposing team offline  
 at a critical time.

The third type of cheating tactics are those that emerge from the nature

of computer games. In tactical shooters, for example, success should depend  
 on the player’s tactics and shooting skill, not on the game mechanics. Yet

there are always shortcomings in the game’s physics model, often introduced by  
 network latency and by the optimisations game designers use to deal with it. For  
 example, you’d normally expect that in a shooting duel, you’d have an advantage  
 if you have the lowest network latency, or if you move ﬁrst. Yet the prediction  
 algorithms used in many game clients cache information about nearby players,  
 so if you leap round a corner, see your enemy and shoot, then the slower your  
 network connection is, the longer it will take him to see you and respond. Mike  
 Bond coined the term ‘neo-tactic’ to refer to players subliminally exploiting such  
 anomalies [280]. That may not of itself be cheating, but in recent years players  
 have started manipulating network connections deliberately to create artiﬁcial  
 lag, whether of incoming packets to delay other players, or our outgoing ones in  
 order to see what other players are about to do.

That brings us on to one of the classic game cheats, namely to have code of

your own for automation and support. People have written a huge variety of  
 tools, from simple routines that repeatedly click a ﬁre button (to hack the games  
 where the rate at which you can physically ﬁre is a factor) through proxies that  
 intercept the incoming network packets, identify the bad guys, examine your  
 outgoing shots, and optimise their aim. These *aimbots* come with different levels  
 of sophistication, from code that does all the target acquisition and shooting, to  
 human-controlled versions that merely improve your aim. They can hook into  
 the packet stream as proxies, into the graphics card, or even into the client code.  
 Another variant on the same theme is the *wall hack*, where a player modiﬁes his

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 762 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

software to see through walls – for example, by changing the graphics software  
 to make them translucent rather than opaque. Such hacks are possible because  
 ﬁrst-person shooters typically send out raw positional information to all players  
 in the game, and leave it up to client software to render it according to the local  
 physics model.

Game companies who sell ﬁrst-person shooters reckon that aimbots and other

client-side hacks seriously spoil other players’ fun, so they use a variety of en-  
 cryption, authentication and DRM mechanisms to reduce not only cheating,  
 but also the perception of cheating – which is almost as damaging to the op-  
 erator [281]. Guard software such as Punkbuster has been around since 2000,  
 using anti-virus techniques to detect attempts to hook into game code or the  
 drivers on which it relies. The large gaming platforms such as Steam have their  
 own DRM mechanisms that attempt to block aimbots and other game cheats,  
 as well as protecting their own revenue by making it harder for customers to  
 resell games [1288]. This is a constant battle, as I discussed in section above,  
 and some techniques such as artiﬁcial lag are difficult to deal with completely.  
 However, gaming is one of the applications in which trustworthy client software,  
 whose protection involves DRM-like mechanisms, has become well entrenched,  
 even though most modern games are locked to customer accounts and most of  
 their logic now runs on a server. The server is also often fortiﬁed with analytics  
 to detect cheating after the event, just like in a professional bridge tournament.

**24.3.5** **Peer-to-peer systems**

From the late 1990s, peer-to-peer ﬁle-sharing became one of the main ways  
 in which music was distributed online. Once people had CD drives on their  
 computers and broadband connections, they could copy and share their favourite  
 tracks. In 1999, Shawn Fanning, an 18-year-old drop-out, revolutionised the  
 music business by creating the Napster service, which enabled people to share  
 MP3 audio ﬁles with each other [1381]. Rather than keeping the ﬁles centrally,  
 which would invite legal action, Napster just provided an index so that someone  
 wanting a given track could ﬁnd out who else had it and was prepared to share  
 or trade. It attracted tens of millions of users, but lawsuits from Hollywood  
 closed it down in September 2002. Systems such as Gnutella and Freenet then  
 borrowed ideas from the world of censorship-resistant systems to set up networks  
 without a central node that could be closed down by legal attacks [439]. These  
 were followed by other systems such as Kazaa and Bittorrent.

I was the designer of an early censorship-resistant system, the Eternity Ser-

vice. The motivation came when an early anonymous remailer, anon.penet.fi,  
 was used to post a message that upset the Scientologists and was closed down  
 after they got a court order forcing its operator to disclose the linkage between  
 users’ real email addresses and the pseudonyms they used on his system [881].  
 The messages that were the subject of the case contained an affidavit by a former  
 minister of their church to the effect that once members had been fully initiated  
 they were told that the rest of the human race was suffering from false con-  
 sciousness; that, in reality, Jesus was the bad guy and Lucifer was the good guy.  
 Well, history has many examples of religions that denounced their competitors  
 as both deluded and wicked; the Scientologists’ innovation was to claim that

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 763 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

their scriptures were their copyright, so the whistleblower’s leak was a breach  
 of copyright. They got away with this argument in a number of jurisdictions  
 until eventually a court in the Netherlands put a stop to it by allowing an NGO  
 there to publish the ‘Fishman affidavit’, as it was called.

The Eternity Service was designed to provide long-term ﬁle storage by dis-

tributing ﬁle fragments across the net, encrypted so that the people hosting them  
 would not be able to tell which fragments they had, and so that reconstruction  
 could only be performed via remailer mechanisms [61]. A later version of this  
 was Publius9, which also provided a censorship-resistant anonymous publishing  
 mechanism [1974].

The United States Copyright Office deﬁnes peer-to-peer networks as net-

works where computers are linked to one another directly rather than through  
 a central server. The absence of a server that can be closed down by court order  
 creates an interesting problem for music industry enforcers. The two tactics on  
 which the music industry relied were suing uploaders and technical attacks on  
 the systems.

One way to attack peer-to-peer systems is to ‘walk the network’ by intro-

ducing a modiﬁed peer, contacting as many other peers as possible, and then  
 identifying them. During the mid-2000s, the music industry tried harassing

users at scale, ﬁling tens of thousands of lawsuits. In many cases people agreed  
 to cease and desist and pay a small penalty rather than ﬁght a case; but in Octo-  
 ber 2007 a federal jury in Duluth, MN., convicted 30-year-old Jammie Thomas  
 of copyright infringement for sharing material on Kazaa and ordered her to pay  
 $9,250 for each of the 24 songs involved in the case. Firms working for the  
 music industry were also uploading damaged music ﬁles to spam out systems  
 (which will usually be legal), and it was suspected that they were also conduct-  
 ing denial-of-service attacks (which in many jurisdictions isn’t). In September  
 2007, a company called Media Defender that worked for the music industry on  
 ‘ﬁle-sharing mitigation’ had several thousand of its internal emails leaked, after  
 an employee forwarded his email to Gmail and his password was compromised.  
 It turned out that Media Defender’s business model was to charge $4,000 per al-  
 bum per month, and $2,000 per track per month, for ‘protection’ that involved  
 attacks on twelve million users of ﬁfteen P2P networks [1501]. Peer-to-peer

systems have also allegedly been attacked by Comcast, which is said to have  
 disrupted its customers’ connections by sending forged reset packets to tear  
 down Bittorrent connections. Comcast might prefer its customers to watch TV  
 over its cable network, so they see its ads, but the allegations raise public policy  
 issues if true: Comcast is not a law-enforcement agency [219].

The state of play in 2020 is that some jurisdictions suffer from this kind of

extortion, from law ﬁrms sometimes referred to as Torrent trolls: in Sweden,  
 for example, there have been tens of thousands of cases where lawyers demand  
 large payments from families claiming that their kids uploaded some copyrighted  
 material [1655]. This appears to be a function of local procedural law more than  
 anything else; in many countries, lawyers can’t be as crooked, or at least not in

9For non-US readers: the revolutionaries Alexander Hamilton, John Jay, and James Madi-

son used the pen name Publius when they wrote the Federalist Papers, a collection of 85  
 articles published in New York State newspapers in 1787–8 and which helped convince New  
 York voters to ratify the United States constitution.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 764 | Ross Anderson |

*24.3. DRM ON GENERAL-PURPOSE COMPUTERS*

this particular way.

In the larger global ecosystem, the big service ﬁrms are now dominant and

the deciding factor in copyright infringement is the notice-and-takedown regime  
 set up under the US DMCA, and followed by similar laws elsewhere. I will

discuss this further in section 24.5.

**24.3.6** **Managing hardware design rights**

Another rights-management ecosystem is the protection of designs licensed for  
 use in hardware. Companies like Arm earn their living by licensing designs for  
 processors and other components that to ﬁrms who make custom chips, whether  
 by designing application-speciﬁc integrated circuits (ASICs) or by using Field-  
 Programmable Gate Arrays (FPGAs).

The ﬁrst use case for hardware protection is when such devices are used to

make it harder to counterfeit products, for example by overrun production. A  
 camera company licenses a circuit that they integrate into a bitstream that’s  
 loaded into an FPGA, that then becomes a key component in a new camera  
 that they have made in a factory in China. They pay for 100,000 licenses, yet  
 200,000 cameras arrive on the market. There are two failure modes: the camera  
 company could have ordered the extra production and lied to the IP owner, or  
 the Chinese factory could be cheating the camera company. In fact, they could  
 both be cheating, each having decided to make an extra 50,000 units. Now

there are technical mechanisms that the camera company could use to stop the  
 factory cheating it, such as personalising each camera with a serial number and  
 so on after manufacture – but these could make it harder to cheat the IP owner.

So the second problem is how the IP owner can tell whether a product con-

tains a particular circuit. The camera company might have licensed a processor  
 or a ﬁlter for one model, then built it into another cheaper model too without  
 declaring it.

These risks cause some large IP vendors to prefer to license their best designs

only to other large ﬁrms, so small startups can be disadvantaged. They also  
 depress sales of FPGAs, whose manufacturers offer mechanisms to tackle the  
 ﬁrst problem by distributing encrypted bitstreams and updates for whole chips;  
 the second problem is harder, because chip design tools come within the trust  
 boundary. Customers need to be able to evaluate designs, and debug designs,  
 which is in tension with controlling dissemination. There has been some use  
 of side-channels for forensics. Owners of semiconductor IP can buy samples of  
 suspect goods, then measure the chips’ precise analog behaviour such as power  
 consumption and timing, which can often reveal the presence of a given func-  
 tional component. Components can even be deliberately designed to generate  
 a suitable signal in their power trace. (Similar techniques are used by military  
 contractors to look for hardware Trojans.)

This brings us to the question of copyright marking.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 765 | Ross Anderson |

*24.4. INFORMATION HIDING*

**24.4** **Information Hiding**

Hollywood’s interest in ﬁnding new mechanisms for protecting copyright came  
 together in the mid-1990s with the military’s interest in unobtrusive commu-  
 nications and public concerns over government efforts to control cryptography,  
 and started to drive rapid developments in the ﬁeld of *information hiding*. This  
 largely refers to techniques for hiding data in other data, such as when a se-  
 cret message is hidden in an MP3 audio ﬁle, or a program’s serial number is  
 embedded in the order in which certain instructions are executed.

Hollywood sought salvation in *copyright marks* embedded unobtrusively in

digital audio, video and artwork. These include *watermarks*, copyright messages  
 which may or may not be hidden but are hard to remove, and *ﬁngerprints* which  
 are hidden serial numbers. For example, when you downloaded an mp3 from  
 Apple’s iTunes music store, it contained a ﬁngerprint embedded in the audio  
 that identiﬁed you. The idea was that if you then uploaded your copy to a ﬁle-  
 sharing system, the copyright owner could sue you. (Some people believed that  
 ﬁngerprinting depressed sales overall because of the legal hazards it created for  
 honest purchasers. Amazon, for example, did not mark MP3 downloads [852].)

The privacy interest is in *steganography* whose purpose is to embed a message

in some cover medium in such a way that its very existence remains undetectable.  
 The conceptual model, proposed by Gus Simmons [1745], is as follows. Alice  
 and Bob are in jail and wish to hatch an escape plan; all their communications  
 pass through the warden, Willie; and if Willie detects any encrypted messages,  
 he will frustrate their plan by throwing them into solitary conﬁnement. So they  
 must ﬁnd some way of hiding their secret messages in an innocuous covertext.  
 As in the related ﬁeld of cryptography, we assume that the mechanism in use  
 is known to the warden, and so the security must depend solely on a secret key  
 that Alice and Bob have somehow managed to share [1753].

There is some similarity with electronic warfare. First, if steganography is

seen as a low-probability-of-intercept communication, then copyright marking  
 is like jam-resistant communication: it may use much the same methods but in  
 order to resist focused attacks it is likely to have a much lower bit rate. We can  
 think of Willie as the pirate who tries to mangle the audio or video signal in such  
 a way as to cause the copyright mark detector to fail. Second, techniques such  
 as direct-sequence spread spectrum that were originally developed for electronic  
 warfare found use in the information hiding community.

Copyright marks don’t have to be hidden to be effective. Some TV stations

embed their logo in a visible but unobtrusive manner in the corner of the picture,  
 and as I noted, academic journal downloads do something similar. However, in  
 what follows I’ll concentrate on hidden copyright marks.

**24.4.1** **Watermarks and copy generation management**

The DVD consortium became concerned that digital video or audio could be  
 decoded to analog format and then redistributed (the so-called ‘analog hole’).  
 They set out to invent a *copy generation management system* that would work  
 even with analog signals. The idea was that a video or music track might be

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 766 | Ross Anderson |

*24.4. INFORMATION HIDING*

unmarked, or marked ‘never copy’, or marked ‘copy once only’; compliant players  
 would not record a video marked ‘never copy’ and when recording one marked  
 ‘copy once only’ would change its mark to ‘never copy’. Commercially sold

videos would be marked ‘never copy’, while TV broadcasts and similar material  
 would be marked ‘copy once only’. In this way, the DVD players available

to consumers would allow unlimited copying of home videos and time-shifted  
 viewing of TV programmes, but could not easily be abused for commercial  
 piracy. The mechanisms depended on hiding copyright marks in the content,  
 and are reviewed in [1167]. For each disk, choose a *ticket X*, which can be a  
 random number, plus copy control information, plus possibly some information  
 unique to the physical medium such as the wobble in the lead-in track. Use a  
 one-way hash function *h* to compute *h*(*X*) and then *h*(*h*(*X*)). Embed *h*(*h*(*X*))  
 in the video as a hidden copyright mark. Have compliant machines look for a  
 watermark, and if they ﬁnd one refuse to play a track unless they are supplied  
 with *h*(*X*) which they check by hashing it and comparing it with the mark.  
 Compliant devices will only record a marked track if given *X*, in which case  
 only *h*(*X*) is written to the new disc. In this way, a ‘copy once only’ track  
 in the original medium becomes a ‘copy no more’ track in the new medium.  
 This ended up in Blu-ray, but that failed in the marketplace, as well as being a  
 complete pain for developers to work with.

Robustness depends on many things including our old friend, the receiver

operating characteristic or ROC, which sets the trade-off between false alarms  
 and missed alarms. It’s not enough for a marking mechanism to have a low  
 missed alarm rate; it needs a low false alarm rate too [1318]. If your player were  
 to detect a ‘no-copy’ mark by mistake in the video you made of your child’s  
 birthday party, then you’d have to buy a pirate player to watch it. So what sort  
 of marks are possible, and how robust are they against forgery, spooﬁng and  
 other attacks?

**24.4.2** **General information hiding techniques**

Information hiding goes back even further than cryptology, having its roots in  
 camouﬂage. Herodotus records tricks used during the wars between the Greeks  
 and the Persians, including hiding a message in the belly of a hare carried by  
 a hunter, tattooing it on the shaven head of a slave whose hair was then al-  
 lowed to grow back, and writing it on the wooden base under the wax of a  
 writing tablet [889]. Francis Bacon proposed a system which embedded a bi-  
 nary message in a book at one bit per letter by alternating between two different  
 fonts [1513]. Until quite modern times, most writers considered hiding conﬁ-  
 dential information much more important than enciphering it [2021]. Military  
 and intelligence organizations are keenly aware that traffic security is often more  
 important than content conﬁdentiality, and have used all sorts of technologies  
 from the microdots used by spies to low-probability-of-intercept radios.

When it comes to hiding data in other data, the modern terminology of the

subject is as follows [1521]. The copyright mark, or in the case of steganography,  
 the *embedded text*, is hidden in the *cover-text* producing the *marked text* or  
 in the case of steganography the *stego-text*. In most cases, additional secret  
 information is used during this process; this is the *marking key* or *stego-key*,

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 767 | Ross Anderson |

*24.4. INFORMATION HIDING*

and some function of it is typically needed to recover the mark or embedded  
 text. Here, the word ‘text’ can be replaced by ‘audio’, ‘video’ and so on, as  
 appropriate.

A wide variety of embedding schemes has been proposed.

*•* Many people have proposed hiding mark or secret message in the least  
 strategy, as the hidden data is easy to detect statistically (the least sig-  
 niﬁcant bits are no longer correlated with the rest of the image), and it’s  
 trivial to remove or replace. It’s also severely damaged by lossy compres-  
 sion techniques.

*•* A better technique is to hide the mark at one or more locations determined  
 receiver had copies of a paper mask with holes cut out of it at random  
 locations. The sender would place his mask over a blank sheet of paper,  
 write his message in the holes, then remove it and compose a cover message  
 including the characters of the secret embedded message. This trick was  
 reinvented in the 16th century by the Italian mathematician Cardan and  
 is now known to cryptographers as the Cardan grille [1001].

*•* A modern version of this hides a mark in a .gif format image as follows.  
 number of pixels. The embedded message is the parity of the color codes  
 for these pixels. In practice even a quite large number of the pixels in an  
 image can have their color changed to that of a similar one in the palette  
 without any visible effects [972]. However, if all the pixels are tweaked in  
 this way, then again the hidden data is easy to remove by just tweaking  
 them again. A better result is obtained if the cover image and embedding  
 method are such that 1% of the pixels can safely be tweaked. Then, if the  
 warden repeats the process but with a different key, a different 1% of the  
 pixels will be tweaked and only 1% of the bits of the hidden data will be  
 corrupted. These can then be recovered using an error-correcting code.

*•* In general, the introduction of noise or distortion – as happens with lossy  
 less of the embedding method unless some kind of error correcting code  
 is added. A system proposed for banknote marking, Patchwork, uses a  
 repetition code – the key selects two subsets of pixels, one of which is  
 marked by increasing the luminosity and the other by decreasing it. This  
 embeds a single bit; the note is either watermarked using that key, or it  
 isn’t [225, 830]. This is reminiscent of differential power analysis: the key  
 tells you how to sort your input data into two piles, and if the key was  
 right they’re noticeably different.

*•* In the general case, one may want to embed more than one bit, and have  
 technique is to use direct-sequence spread-spectrum techniques borrowed  
 from electronic warfare [1890]. You have a number of secret sequences,  
 each coding a particular symbol, and you add one of them to the content  
 to mark it.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 768 | Ross Anderson |

*24.4. INFORMATION HIDING*

*•* Spread spectrum encoding is often done in a transform space to make its  
 pression. These techniques are also commonly used in conjunction with  
 perceptual ﬁltering, which emphasises the encoding in the noisiest or per-  
 ceptually most signiﬁcant parts of the image or music track, where it will  
 be least obtrusive, and de-emphasises it in quiet passages of music or large  
 expanses of color [288].

*•* Some schemes use the characteristics of particular media, such as a scheme  
 hundredth of an inch [315], or adding extra echoes to music below the  
 threshold of perception [225]. So far, such techniques don’t seem to have  
 become as robust, or as widely used, as generic techniques based on keyed  
 embedding using transform spaces, spread spectrum and perceptual ﬁlter-  
 ing.

Progress in copyright marking was very rapid in the late 1990s: people in-

vented marking schemes which other people broke, until some systems were  
 adopted in banknotes and in some tools such as Adobe’s. From the mid-2000s,  
 interest in copyright marking waned with the move to broadband, but research  
 in steganography and steganalysis continued, merging with research in image  
 forensics.

**24.4.3** **Attacks on copyright marking schemes**

Throughout this book, we’ve seen attacks on cryptographic systems that occa-  
 sionally involved cryptanalysis but more often relied on mistaken assumptions,  
 protecting the wrong things, protocol failures and implementation bugs. And  
 in the history of technology as a whole, inventions tend to end up being used to  
 solve problems somewhat different from the problems the inventor was originally  
 thinking about. Copyright marking has been no different on either count.

*•* In the beginning, many people tackled the problem of embedding hidden  
 But this is a non-problem. Lawyers almost never have any difficulty in  
 proving ownership of an exhibit; they don’t rely on technical measures  
 which might confuse a jury, but on documents such as contracts with  
 bands and model release forms.

*•* As usual, many designers ignored Kerckhoffs’ principle – that the security  
 But when marks are used to prove whether a particular digital object was  
 licensed, this means disclosing them in court together with the marking  
 keys, so it may be necessary to use multiple keys.

*•* As an example, color copiers sold in the US hide a *Machine Identiﬁcation*  
 currency forgers [2002]. Introduced by Xerox and Canon in the 1980s,  
 apparently following a secret agreement with one or more governments,  
 its existence was disclosed in a court case in the Netherlands in 2004. The

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 769 | Ross Anderson |

*24.4. INFORMATION HIDING*

mechanism was then reverse engineered in a crowdsourced effort led by  
 EFF. The MIC is a pattern of yellow dots 0.1mm in diameter that is barely  
 visible to the human eye and repeated about 150 times on an A4 colour  
 copy. There is now software to identify and remove it, so whistleblowers  
 can sanitise sensitive documents before leaking them to the press [1602].

*•* Many marks simply add some noise to the signal. But if all the frames in  
 then subtract it out. Or you supply some known content to a marking  
 system, and compare its input and output. Even if the mark is applied in  
 a tamper-resistant process immediately after decryption, and every device  
 adds a different mark, then if the mark consists of small signals added  
 at discrete points in the content, an opponent can just decrypt the same  
 ciphertext with several different devices and compare them to remove the  
 marks.

*•* There have been attempts to develop a marking equivalent of public-key  
 only one principal could detect, or anyone could detect a mark that only  
 one principal could have inserted. The former seems just about feasible  
 if the mark can be inserted as the cover audio or video is being manufac-  
 tured [494]. The latter seems a lot harder. First, you can’t authenticate  
 all of an image by embedding a signature in it, as then you’d be modifying  
 it in order to prove that it has not been modiﬁed. Second, if you try to  
 authenticate just the high-order bits or the salient features, then there  
 are robustness issues: given a device that will detect a mark, an attacker  
 can remove it by applying small changes to the image until the decoder  
 cannot ﬁnd it anymore [1511, 1168], then apply their own signature. So  
 the main effort was invested in mechanisms that put a different mark in  
 each instance of the content, as it is decrypted.

*•* Steganalysis techniques were developed to break most embedding schemes.  
 nisms at the Information Hiding Workshop, and the following year they’d  
 be broken. The most proliﬁc attack team was Jessica Fridrich and her  
 students at Binghamton; her book on steganography is the starting point  
 for serious work on the subject [724].

*•* The most successful marking startup – Digimarc – set up a service to track  
 embed invisible ﬁngerprints, and had a bot which crawled the web looking  
 for marked pictures and reported them to the copyright owner. There were  
 various ways to defeat this. For example, a marked image could often be  
 chopped up into smaller images which together look just like the original  
 when rendered on a web page but in which a copyright mark won’t be  
 detected (Figure 24.6) [1516]. Digimarc worked for a while on monitoring  
 broadcast streams; but over time, AI improved to the point that software  
 can identify which song is being played directly. Digimarc moved into

security printing, licensing their marking technology to central banks as a  
 counterfeit detection measure. For example, it’s found in Euro banknotes,

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 770 | Ross Anderson |

*24.4. INFORMATION HIDING*

Figure 24.6: – the Mosaic attack (courtesy Jet Photographic,

www.jetphotographic.com)

which it prevents from being scanned or copied using the latest equip-  
 ment [2061]. Software packages such as Photoshop and Paintshop Pro

now refuse to handle marked images. Digimarc now monitors packaging  
 and provides labeling systems.

*•* The most general attacks on imperceptible copyright marking schemes  
 domly duplicating or deleting sound samples to introduce inaudible jitter;  
 techniques used for click removal and resampling are also powerful mark re-  
 movers. For images, a tool my students developed, called Stirmark, intro-  
 duces the same kind of errors into an image as printing it on a high quality  
 printer and then scanning it again with a high quality scanner. It applies  
 a minor geometric distortion: the image is slightly stretched, sheared,  
 shifted, and/or rotated by an unnoticeable random amount This defeated  
 almost all the marking schemes in existence when it was developed and is  
 now a standard benchmark for copyright mark robustness [1516].

For a fuller account of attacks on copyright marking schemes, see [724].

It’s still difficult to design marking schemes that remain robust once the mark  
 detection algorithm is known.

Perhaps the key technical factor that killed copyright marking wasn’t an at-

tack but latency. This is really important for streamed sports events; you don’t  
 want to hear cheering from next door before you see the goal. Recently media  
 streaming standards (DASH, HLS) have been updated to support downloading  
 media chunks before they have been written completely to ‘ﬁx’ this. Apparently  
 server-side watermarking to identify who re-streamed a stream can introduce a  
 lot of latency. This helped drive the adoption of direct recognition of infringing  
 material instead. One pioneer, Shazam, was bought by Apple in 2017; Google

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 771 | Ross Anderson |

*24.5. POLICY*

developed its own Content ID for YouTube with a database of content ﬁnger-  
 prints with information about where copyright has been claimed, and when  
 videos are uploaded or live streamed they are looked up in this database. Copy-  
 right owners can opt to monetize the video by getting a share of ad revenue,  
 or block it. Similar technology is used to block content that’s objectionable for  
 other reasons: child sex abuse material is mostly recognised using a Microsoft  
 system called PhotoDNA.

**24.5** **Policy**

There was a vigorous policy debate in the 1990s and 2000s between the tech  
 industry and many of the owners of ‘intellectual property’ (IP) – copyright,  
 patents and trademarks – as the opening up of the Internet made copying easy  
 and threatened traditional music, book and ﬁlm publishers10. The reaction in-  
 cluded a series of laws from copyright term extension through America’s Digital  
 Millennium Copyright Act (DMCA) to an IP Enforcement Directive in Europe,  
 which shifted power in ways that many people in tech and elsewhere felt to be  
 threatening. The get-out for tech was section 230 of the US Communications  
 Decency Act of 1996 (CDA) which states that ‘No provider or user of an in-  
 teractive computer service shall be treated as the publisher or speaker of any  
 information provided by another information content provider’ so platforms can-  
 not be held liable for copyright infringement by users. This favoured the growth  
 of information service ﬁrms in the USA rather than Europe.

The US DMCA does give copyright owners the power (‘Notice and Take

Down’) to compel ISPs to take down websites with infringing material. Al-

though there is also a provision (‘Notice and Put Back’) for the subscriber to  
 ﬁle a counter notice and have his stuff put back within 14 days unless the copy-  
 right owner ﬁles suit, in practice many ISPs will just terminate a customer’s  
 service rather than get involved in litigation. This led not just to a lot of mu-  
 sic copying using peer-to-peer systems, but to ﬂoods of takedown requests from  
 music industry lawyers, as well as to the push for DRM that we discussed earlier.

Over half of the takedown requests to Google come from the top 16 copyright

owners, with the top three generating over a billion a year – many of them to  
 links that are not even on Google. Many complaining organisations get few or  
 none of the links they complain about removed, as they are either not relevant or  
 judged to be non-infringing; see Google’s transparency reports for details [800].  
 This has real policy consequences: censoring a Chinese shop that pretends to  
 be Nike is one thing, while censoring Black Lives Matter Peckham in response  
 to a complaint from a white supremacist is quite another.

There are many side-effects: for example, the legal rules that allowed copy-

ing for personal use (‘fair use’ in the USA and ‘fair dealing’ in the UK) are being  
 replaced by technical controls that don’t. For example, when I applied for plan-  
 ning permission to extend my kitchen, I had to ﬁle four copies of a local plan;

10The term ‘intellectual property’ is controversial. Many activists object to it as a pro-

paganda term coined by corporate lobbyists who want people to start seeing patents and  
 copyrights as permanent natural rights, like title to real estate or human rights, rather than  
 as the temporary monopolies that they are.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 772 | Ross Anderson |

*24.5. POLICY*

but the map software at our university library only lets you print three copies.  
 This is an entirely deliberate act by the Ordnance Survey to maximise its rev-  
 enue. Legal controls are supplemented by access controls, and the legal privilege  
 given to those access controls by the DMCA and comparable EU laws creates a  
 new bundle of rights, described by legal scholars as ‘paracopyright’ [532].

In effect, copyright regulations are no longer made by lawmakers in Washing-

ton or Brussels, but by programmers working for Microsoft or Apple or Amazon.  
 The result has been to erode the rights of copyright users. In one spectacular ex-  
 ample, Amazon quietly removed from its customers’ Kindles an edition of George  
 Orwell’s ‘1984’ and ‘Animal Farm’ over which some dispute had arisen [1831].  
 This was a sobering reminder of the huge gap between owning a physical copy  
 of a book, and ‘owning’ an e-book – in fact you just bought a license from a  
 vendor who wrote the license so as to give you next to no rights at all.

At the same time, copyright law suddenly became relevant to millions of

people. Whereas in the past it was only a concern of specialists such as pub-  
 lishers, it now touches the lives of everyone who downloads music, time-shifts  
 movies, or maintains a personal web page. As the law has failed to keep up  
 with technology, the gap between what it permits and what people actually do  
 has become wider. In the UK, for example, it’s technically illegal to rip a CD  
 to listen to on your phone; yet as this is one of the main reasons that people  
 still buy CDs, the British Phonographic Industry (the trade body) graciously  
 says it won’t sue anybody. But many of the minor infringements that used to  
 take place in private, or unsurveilled public spaces (such as singing a song in a  
 pub), now go online (as when a phone video clip of the song gets on someone’s  
 social-network page). John Tehranian calculates that a typical law professor  
 commits over 80 copyright infringements a day, carrying statutory penalties of  
 over $10m [1866]. In effect, we only tolerated copyright law because it wasn’t  
 enforced against private individuals. Technology makes enforcement possible,  
 the consolidation of copyrights into an ever smaller number of corporate own-  
 ers and collecting societies makes for a concentrated lobby, greed makes abuses  
 happen, and the frictions increase.

The consolidation of copyrights also leads to injustice in the distribution of

income. I already mentioned the problems with collecting societies, which in  
 effect tax venues and distribute the proceeds in such a way that the rich get  
 lots and the small fry not so much at all; this has become worse with streaming,  
 whose payouts are a function of plays rather than users. So if my granddaughter  
 pays £10 a month and listens to Ariana Grande four hours a day while I pay the  
 same and listen to Kathryn Tickell two hours a week, then rather than giving  
 them £10 each (less Apple’s 30% commission), Ariana will get fourteen times  
 what Kathryn gets [1553]. This means that most of your subscription – or at  
 least of the money the tech ﬁrms don’t take one way or another – goes to the  
 megastars like Ariana, and Ed Sheeran and Lady Gaga.

There are also privacy concerns. In the old days, people would buy a book or

a record for cash; the move to downloads means that servers run by ﬁrms such  
 as Google, Spotify and Apple have a record of what people watch and listen  
 to, and this can be subpoena’ed. (The move to online bookselling and then to  
 Kindles has created similar records at Amazon.) These records are also used  
 for marketing. A survey for the Privacy Commissioner of Canada found many

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 773 | Ross Anderson |

*24.5. POLICY*

examples of intrusive behavior, including e-book software proﬁling individuals,  
 DoubleClick advertising in a library service, systems tracking individuals via IP  
 addresses, and contradictions between vendors’ stated privacy policies and ob-  
 served behaviour – including undisclosed communications to third parties [682].  
 Why do copyright owners, or big tech ﬁrms claiming to act on their behalf, get  
 away with so much? The answer lies in the dynamics of lobbying.

**24.5.1** **The IP lobby**

The IP lobby has its modern origins in an effort by the drug company Pﬁzer to  
 extend patent protection on its drugs from the USA to less developed countries  
 like Brazil and India in the 1970s. The history is told by Peter Drahos and  
 John Braithwaite [581]; in summary, Pﬁzer and the other drug companies allied  
 themselves with the music and ﬁlm industry (who wanted to cut bootlegging  
 and copying), the luxury-goods industry (who wanted to reduce the number of  
 cheap knock-offs), and a number of other US players (including the Business  
 Software Alliance), and persuaded the US government to start applying pres-  
 sure on other countries to bring their patent, copyright and trade-mark laws  
 in line with America’s. From the mid-1980s this was largely a matter of bul-  
 lying less developed countries who wanted trade deals, but in 1995 a treaty on  
 Trade-Related Aspects of Intellectual Property Rights (TRIPS) took effect for  
 members of the World Trade Organisation (WTO), followed by two treaties of  
 the World Intellectual Property Organisation (WIPO) in 1996. Essentially the  
 USA and the EU got together and bullied holdouts like India and Brazil.

The implementation of these treaties stirred up opposition in developed coun-

tries as people began to realise how they might be affected. In the USA, the  
 Digital Millennium Copyright Act of 1998 made it an offence to circumvent a  
 copyright-protection mechanism, as required by WIPO, while in the European  
 Union the Copyright Directive of 2001 had a similar effect. This was seen as  
 enabling vendors to create closed platforms and control competition; it was also  
 seen as a threat by the free and open source software movement, and by secu-  
 rity researchers – especially after the Russian researcher Dmitri Sklyarov was  
 arrested at a US conference at the request of Adobe, after his employer had sold  
 tools circumventing password protection on PDF documents.

There were many other high-proﬁle incidents; for example, I was on the

program committee of the 2001 Information Hiding Workshop when an attempt  
 was made by the Recording Industry Association of America (RIAA) to force  
 the program chair to pull a paper by Ed Felten and his students describing  
 vulnerabilities in a copyright marking scheme being touted for a digital music  
 standard [495]. Ed then sued RIAA, in a landmark academic-freedom case [620].  
 The irony is that the promoters of this scheme had issued a public challenge  
 to academics and others to break it. The next case was Bunnie Huang’s book  
 “Hacking the Xbox”: this described how, as an MIT student, he’d overcome the  
 protection mechanisms in the ﬁrst version of Microsoft’s games console [930].  
 The book he wrote caused his publisher to take fright, but he found another one.  
 The encroachment on liberties threatened by rights-management mechanisms  
 and anti-hacking laws led to the growth of digital-rights NGOs in a number of  
 countries (others had them already as a result of the ‘Crypto Wars’; I’ll discuss

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 774 | Ross Anderson |

*24.5. POLICY*

all this in more detail in section 26.2.7).

One turning point came in 2003–4, as the IP lobby was trying to steer a

further measure through Brussels, the IP Enforcement Directive. In its original  
 form, this would have further ratcheted up the penalties on infringers and re-  
 moved the prospects for public-interest defences based on free speech or fair use.  
 This time opponents of the measure managed to assemble a sufficiently strong  
 coalition of opposing interests that the measure was substantially amended.  
 This opposition led to the establishment the following year of EDRi, an NGO  
 that promotes European digital rights, and is supported by several dozen NGOs  
 in Europe who realised that a lobbying presence in Brussels was essential.

The IP lobby’s mistake was trying to compel every country in Europe to

make patent infringement a crime, rather than just a civil matter. This was  
 intended by Big Pharma to undermine ﬁrms who make low-cost generic ver-  
 sions of drugs once they have come off patent. At present, drug patent holders  
 try to prolong their patents by ‘evergreening’ – ﬁling subsidiary, later patents,  
 with often dubious derivative claims – which the generic drugmakers deal with  
 by offering their distributors indemnities against having to pay damages. Mak-  
 ing infringement a criminal matter would have upset these arrangements. This  
 caused the generic drugmakers to oppose the directive vigorously, along with su-  
 permarkets, car parts dealers and consumer groups. Even the software industry  
 started to get nervous: we pointed out to Microsoft that thousands of companies  
 believe that Microsoft is infringing their patents, but don’t have the money to go  
 the distance in a civil court. If patent infringement became a crime, surely they  
 would take their grievances to the police? Would Bill risk arrest on some future  
 trip to Europe? The attempt to criminalise patent infringement collapsed when  
 tech ﬁrms withdrew their support. A rich, powerful lobby isn’t stopped by ﬁne  
 words, or by outrage from university professors and free-software activists. It’s  
 stopped when it comes up against another rich, powerful lobby pushing in the  
 opposite direction.

Some copyright activists hope that once copyright expires – or assuming that

lots of material can be made available under a Creative Commons license – then  
 everything will be hunky-dory. I doubt it. The theory behind both copyright  
 and patent was to offer creators a temporary monopoly in order to increase  
 the supply of creations. Initially copyright was for 18 years, then 35, then 50,  
 then the creator’s lifetime plus 70 years after that. Cynics noted that whenever  
 Mickey Mouse was in danger of going out of copyright, the US government would  
 step in to increase the copyright term, and bully other governments to fall in  
 line. (Other cynics noted that the copyright term for musical performance was  
 extended from 50 years to 70 after Sir Cliff Richard let the then Prime Minister  
 Tony Blair holiday at his mansion in Barbados.) Some lawyers would like to  
 extend copyright term indeﬁnitely, but that violates the social contract on which  
 copyright is based and it also doesn’t solve the problem of preservation: many  
 publishers have failed to look after their own back catalogue properly and had  
 to retrieve copies from national deposit collections.

Curating old bits costs money, just as curating old manuscripts does; indeed

the ﬁlm industry has recently discovered that archiving digital productions ac-  
 tually costs more than they used to pay in the old days, when they just locked  
 away the master copies in an old salt mine. There’s just an awful lot of bits

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 775 | Ross Anderson |

*24.5. POLICY*

generated during digital production, and copying them to new disks every few  
 years isn’t cheap. In the long term, once bitstrings belong to nobody, who will  
 pay for their upkeep? Might we extend the existing taxpayer-funded deposit li-  
 brary system to digital materials? But such organisations typically fail to make  
 much progress with digital materials for a number of reasons, from lack of un-  
 derstanding to being too defensive about copyright law11. There has been a very  
 creditable effort by the Internet Archive, a San Francisco NGO, to preserve on-  
 line material for future generations, and it has run an open library project since  
 2006. Google scanned many books in university libraries, eventually getting a  
 legal settlement with authors and other interested parties following a long court  
 case12. As a result, Google Books can make millions of volumes searchable, and  
 supply the full contents of books that are out of copyright. Where a books is  
 still in copyright, it can let people search and see snippets as a fair use allowed  
 under copyright law, but it cannot sell an electronic version without the pub-  
 lisher’s agreement. (It had wanted to sell electronic versions of everything and  
 simply pay the publishers a ﬁxed royalty, so as to challenge Amazon’s hold on  
 the book market.) The latest development in 2020 is a lawsuit by book publish-  
 ers (including Wiley, the publisher of this book) to stop the Internet Archive  
 lending out electronic copies of books [1000]. The copyright wars drag on, even  
 despite the pandemic.

**24.5.2** **Who beneﬁts?**

As I mentioned in section 8.6.4, a turning point in the copyright wars came in  
 2005. In January of that year, Google’s chief economist Hal Varian addressed  
 a DRM conference in Berlin and asked who would beneﬁt from stronger DRM.  
 He pointed out that, in classical economic theory, a technical link between two  
 industries would usually beneﬁt the more concentrated industry (for example,  
 car makers and car parts). But the platform industry was concentrated (then  
 it was Apple, Microsoft and Sony) while the music industry was less so (four  
 majors and many independents): so why should the music industry expect to  
 be the winners from better DRM? Economic theory says that platform vendors  
 should win more. The music industry scoffed, and yet by the end of that year  
 they were hurting – by the fall of that year, they were tearfully lobbying the  
 UK government and the European Commission to ‘do something’ about Apple,  
 such as forcing it to open its FairPlay DRM scheme.

Over the next few years, Hal’s prediction came true. The music majors

lost their market power to ﬁrms like Apple, Amazon and Spotify, while Netﬂix  
 established a dominant position in distributing video. Music downloading – with  
 or without DRM – changed the structure and dynamics of the music industry.  
 Bands used to rely on the majors to promote them, but now they can do that  
 themselves by giving away their albums on their websites; they always made

11When the British Library wanted to archive our NGO web page they wanted us to sign

copyright release and indemnity forms, which we couldn’t do for material from third parties  
 or written by people who’d left or died. The only practical way forward is to just put stuff  
 online and take it down if anyone makes a convincing objection. That’s what tech ﬁrms do;  
 legacy organisations often don’t have the conﬁdence.

12The Authors Guild, Inc. et al v. Google, Inc.; October 16, 2015 (2d Circuit); November

14, 2013 (SDNY).

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 776 | Ross Anderson |

*24.6. ACCESSORY CONTROL*

most of their money from performances, and now they make more than ever –  
 just as John Perry Barlow had predicted back in 1994. In fact, smart bands now  
 go with an indie label, as then they’ll get a bigger share of the streaming and  
 other revenues. And thanks to the pandemic, there is now a rapidly-growing  
 new sector of online concerts, where bands perform in empty venues and stream  
 live to their fans, cutting out both the subscription streaming services and the  
 big ﬁrms that own the big venues [1685].

**24.6** **Accessory Control**

One of the most important and rapidly-growing uses of cryptographic mecha-  
 nisms and of rights-management technology generally is in accessory control.

The story starts in 1895 when King Camp Gillette invented the disposable

razor blade, and subsidised razors from later sales of blades. Economists call this  
 strategy *two-part pricing*, or even just the ‘razors and blades’ model, in Gillette’s  
 memory. The tech industry ﬁrst adopted it for games consoles; it was then

adopted by printer makers who subsidise the printers from the ink cartridges,  
 starting in 1996 with the Xerox N24 (see [1822] for the history of cartridge chips).  
 In a typical system, if the printer senses a third-party cartridge, or a reﬁlled  
 cartridge, it may silently downgrade from 1200 dpi to 300 dpi, or even refuse to  
 work at all. In 2003, expiry dates and ink usage controls were added [1207]; and  
 modern cartridges now limit the amount of ink dispensed electronically rather  
 than waiting for it to run out physically. The latest development is region

coding: you can’t use US ink cartridges in a recently UK-purchased HP printer.  
 Other industries are adopting this technology. For example, the amount of RAM  
 you are allowed to use in our lab oscilloscope depends on how much you paid  
 for it.

After some grumbling, European regulators decided to put up with this,

but in the USA, the matter was decided in court. The printer maker Lexmark  
 sued SCC, which had reverse-engineered their print-cartridge crypto, alleging  
 violation of the Digital Millennium Copyright Act. Although they won at ﬁrst  
 instance, they lost on appeal in 2004 [1157]. In a similar case, Chamberlain  
 (who make garage door openers) sued Skylink (who made compatible openers)  
 and also lost, losing the appeal too in 2004. This settled US law in favour of  
 a free market for cryptologists, which was the position before the DMCA came  
 along [1647]. A ﬁrm wanting to control its aftermarket using crypto chips is free  
 to hire the smartest cryptographers it can ﬁnd to build authentication chips  
 that are really hard to hack, while its competitors are free to hire the smartest  
 cryptanalysts they can ﬁnd to try to reverse-engineer them.

There are many, many more examples. Even things that never used to

have electronics in them, and that don’t need electronics for any purpose, have  
 acquired chips to enforce predatory business models. There are hundreds of  
 examples: one that came up in 2020 as I was revising this chapter is their use  
 in water ﬁlters in GE fridges. Six months after he bought a ‘smart’ fridge, Jack  
 Busch got a demand that he buy another water ﬁlter for $54.99. It turned out  
 that the ﬁltered water option would turn itself off unless you bought a new ﬁlter  
 every six months, whether you needed it or not. Jack duly ﬁgured out a hack

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 777 | Ross Anderson |

*24.6. ACCESSORY CONTROL*

and published it [353].

Is accessory control objectionable? The view that I took in the second edi-

tion of this book was that of standard economics: depends on how competitive  
 the markets are. If ink cartridges have a high proﬁt margin but the market  
 for printers is competitive, competition will push down the price of printers to  
 compensate for the high-priced cartridges [1942]. But in many other industries  
 it might be anticompetitive; it just depends on how concentrated the industry  
 is, and in winner-take-all platform markets it could be particularly objection-  
 able [73].

I have since changed my mind. Competition matters, and we’re seeing less

of it as one industry after another adopts software in its products and becomes  
 more like the software industry, with the tendency to monopoly that we dis-  
 cussed in Chapter 8. For example, John Deere now ﬁts its tractors with locks  
 that limit repairs to authorised dealers, causing great resentment among farmers  
 at having to pay a $230 call-out and $135 an hour for a technician to authorise  
 a spare part [1070]. The use of cryptographic mechanisms for product tying and  
 bundling is among the anti-competitive factors with which our policymakers are  
 now realising they have to deal. In the case of tractors, a right-to-repair law  
 may be one of the necessary mitigations.

Sustainability also matters, and technical tying mechanisms are often about

shortening product lives, leading to unnecessary consumption. Forcing a six-  
 monthly change of water ﬁlter cartridges is a good example; we use ours for  
 about ﬁve years. Such mechanisms also lead to products that are fragile and  
 difficult to maintain. Another common outcome if you buy a ‘smart fridge’ is  
 that it will turn into a frosty brick a couple of years later, when the vendor stops  
 maintaining the server that it speaks to. I will discuss this at greater length in  
 section 28.5.

The covid pandemic has illustrated other side-effects of accessory control.

Early in the lockdown, some hospitals didn’t have enough batteries for the  
 respirators used by their intensive-care clinicians, now they were being used  
 24 x 7 rather than occasionally. The market-leading 3M respirators and the  
 batteries that powered them had authentication chips, so the company could  
 sell batteries for over $200 that cost $5 to make. Hospitals would happily have  
 bought more for $200, but China had nationalised the factory the previous  
 month, and 3M wouldn’t release the keys to other component suppliers. The  
 ﬁx in this case was indeed competition. Respirators from other suppliers are  
 cheaper and don’t insist on proprietary batteries, while in Southampton, Paul  
 Elkington and colleagues at the medical school designed their own respirator,  
 making the design open to everyone in the world who wants to make them [623].  
 With luck 3M will lose the dominant market position they abused, but there  
 was a real cost to clinical staff who didn’t have enough personal protective  
 equipment in the early months of the pandemic. Market-control mechanisms  
 can have implications not just for sustainability tomorrow, but for safety today.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 778 | Ross Anderson |

*24.7. SUMMARY*

**24.7** **Summary**

The technical protection of digital content against unauthorised copying is a  
 wicked problem both technically and politically. It’s difficult technically because  
 general-purpose computers can copy bitstrings at no cost, and it’s difficult politi-  
 cally because rights-management technology has done a lot of collateral damage.  
 That the music industry itself was one of the casualties may have been just, but  
 doesn’t solve the continuing problems. These are tied up with much broader  
 and deeper problems of competition, consumer protection and sustainability.

**Research Problems**

Many of the tough problems around copyright in 2020 are policy problems rather  
 than technical ones. If you want to do technical work on information hiding or  
 digital image forensics, you might read Jessica Fridrich’s books as a starting  
 point [724]. For software obfuscation, you might start with the report of a

2019 Dagstuhl seminar on the subject organised by Bjorn De Sutter and col-  
 leagues [555]. One open problem that spans both technology and policy is the  
 privacy of the anti-cheat engines used in computer games. What information  
 do they collect from your PC, where do they send it, and is this reasonable? Is  
 it even legal?

**Further Reading**

Kahn is, as usual, good historical background reading [1001]. The software

copy protection techniques of the PC era are discussed in [829]; there’s a his-  
 tory of pay-TV systems in [1255]. As for information hiding, there’s a book  
 by Katzenbeisser and Petitcolas [1023], as well as Jessica Fridrich’s books [724].  
 The standard reference on game security is by Greg Hoglund and Gary Mc-  
 Graw [912]; see also Jeff Yan and Brian Randell for the history of computer  
 game cheating [2054, 2056].

For a principled discussion of the policy issues around copyright and open cul-

ture, you might start with Pam Samuelson [1646, 1647] and Larry Lessig [1144,  
 1145]. Then I’d suggest you read up on whatever application areas are rele-  
 vant to you. If you’re an academic, you ought to read up about the tragedy of  
 Aaron Swartz – the founder of Reddit who killed himself after putting millions  
 of scientiﬁc papers online and being hounded by publishers’s lawyers – and the  
 long-running battles around Sci-Hub, which makes scientiﬁc papers available to  
 all in deﬁance of copyright. If you play music for money you may want to follow  
 the tussles around streaming and the antitrust settlement between Live Nation  
 and Ticketmaster. If you play music in pub sessions you might be interested in  
 the controversy around the Irish Music Rights Organisation.

If you’re a lawyer or policymaker, you would do well to talk to NGOs en-

gaged on copyright issues. Here for example is the view of European Digital  
 Rights (EDRi): “In the digital environment, citizens face disproportionate en-  
 forcement measures from states, arbitrary privatised enforcement measures from

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 779 | Ross Anderson |

*24.7. SUMMARY*

companies and a lack of innovative offers, all of which reinforce the impression  
 of a failed and illegitimate legal framework that undermines the relationship  
 between creators and the society they live in. Copyright needs to be funda-  
 mentally reformed to be ﬁt for purpose, predictable for creators, ﬂexible and  
 credible.”

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 780 | Ross Anderson |