**Chapter 28**

**Assurance and**  
 **Sustainability**

**There are two ways of constructing a software design. One way is to**

**make it so simple that there are obviously no deﬁciencies.**

**And the other way is to make it so complicated that**

**there are no obvious deﬁciences.**

– Tony Hoare

**Security engineers are the litigation lawyers of tech.**

**We only get paid when something is wrong**

**and we can always ﬁnd something wrong.**

– Dave Weston

**To improve is to change; to be perfect is to change often.**

– Winston Churchill

**28.1** **Introduction**

I’ve covered a lot of material in this book, some of it quite tricky. But I’ve left  
 the hardest parts to the last. These are the questions of *assurance* – whether  
 the system will work; its cousin *compliance* – how you satisfy other people about  
 this; and *sustainability* – how long it will keep on working. How do you decide to  
 ship the product? How do you sell the security and safety case to your insurers?  
 How long are you going to have to maintain it, and at what cost?

What’s new in 2020 is *sustainability*. In the 2008 edition, I called this chapter

‘Evaluation and Assurance’, and ended up by remarking that sound processes for  
 vulnerability disclosure and product update were beginning to be as important  
 as pre-market testing. The emphasis back then was on testing and evaluation  
 schemes like the Common Criteria. That world is now moribund: the idea that  
 a device should be secure because someone spent $100,000 getting an evalua-  
 tion lab to test it ﬁve years ago would strike most people nowadays as quaint.

911

*28.1. INTRODUCTION*

Assurance is no longer static.

Ten years ago, we knew how to make two types of secure system. We had

things like phones and laptops, which contained software and were online, but  
 were sort-of secure because the software got patched once a month. And we  
 had things like cars and medical devices, which contained software but were not  
 online; you tested them to death before they were put on sale, and then hoped  
 for the best, as patching meant a physical recall. Now we’ve started to put cars  
 and medical devices online, so they have to be patched online too.

The number of vulnerabilities reported in common platforms is so great that

we have to automate the process. As we described in the previous chapter,

the software development lifecycle has become DevOps and then DevSecOps;  
 the online components of systems are maintained using continuous integration,  
 while components in the ﬁeld need regular upgrades.

With a new product, assurance can be measured roughly by whether capable

motivated people have beat up on the system enough. But how do you deﬁne  
 ‘enough’? And how do you deﬁne the ‘system’? How do you deal with people  
 who protect the wrong thing? And how do you deal with usability? Too many  
 systems are designed for use by alert experienced professionals, but are too  
 tricky for ordinary folk or are intolerant of error. Once they get ﬁelded, the  
 injury claims or fraud disputes start to roll in.

In the security engineering of a decade ago, we often talked of assurance

in terms of *evaluation*, which was about how you assembled the evidence to  
 convince your boss, your clients, and (if need be) a jury, that it did indeed work  
 (or that it did work at some particular time in the past). As we’ve seen again  
 and again, things often fail because one principal carries the cost of protection  
 while another carries the risk of failure. Third-party evaluation schemes such  
 as the Common Criteria were supposed to make these risks more transparent  
 and mitigate them, but ended up acting as a liability shield – particularly in the  
 public sector and in regulated industries such as banking. Systems protecting  
 classiﬁed information were subjected to extensive compliance requirements and  
 had to use evaluated products at the attack surface; much the same held, with  
 different details, for payment systems. Evaluation was driven by compliance.

Compliance is still the main driver of security design and investment, but

it places much less emphasis on requiring evaluated products at speciﬁc trust  
 boundaries. The details vary from one industry to another. When we look at  
 medical systems, cars or aircraft we ﬁnd regulatory regimes driven by safety that  
 are starting to incorporate security. General business systems have policy set  
 by the Big Four audit ﬁrms, and payment systems by PCI. We have touched on  
 some of their speciﬁc requirements in previous chapters; there are some broader  
 issues and principles that we’ll try to pull together here.

Right at the start of this book, in Figure 1.1, I presented a framework for

security engineering based on incentives, policy, mechanism and assurance.

*• Incentives* are critical, as we’ve seen time and again. They often fall outside  
 ment within which the security policy has to be deﬁned.

*• Policy* is often neglected, as we’ve seen: people often end up protecting

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 912 | Ross Anderson |

*28.1. INTRODUCTION*

the wrong things, or protecting the right things in the wrong way. We  
 spent much of Part II of the book exploring security policies for different  
 applications.

*• Mechanisms* may be independent of policy, but can interact with it by

*• Assurance* is our estimate of the likelihood that a system will not fail in a  
 as the process used to develop and maintain it; the people who develop  
 and maintain it; and speciﬁc technical assessments, such as the statistics  
 of failure rates, bug reports, breach reports and insurance claims. It was  
 traditionally about *evaluation* – whether, given the agreed security policy  
 and strength of mechanisms, a product had been implemented correctly.  
 Had the bugs been found and ﬁxed? Could you quantify the mean time to  
 failure? Nowadays it’s increasingly about the vendor’s future commitment.  
 For how long, and how diligently, will the system be patched?

By the second edition of this book in 2008, I noted that the big missing

factor was usability. Most system failures have a signiﬁcant human component.  
 Usability is a cross-cutting issue in the above framework: if done properly, it has  
 a subtle effect on policy, a large effect on choice of mechanisms, and a huge effect  
 on how systems are tested. It cuts across individual products: a common reason  
 for accidents is that different products have different user interfaces, an issue to  
 which we’ll return later. However, designers often saw assurance simply as an  
 absence of obvious bugs, and designed technical protection mechanisms without  
 stopping to consider human frailty. (There are some exceptions: bookkeeping  
 systems are designed to cope with both error and fraud.)

Usability is not purely a matter for end-users, but for developers too. Many

vulnerabilities arise because security mechanisms are too hard to understand or  
 too ﬁddly to use. Developers often didn’t use operating-system access controls,  
 but just ran their code with administrator privilege instead; when mobile phones  
 didn’t allow this, they kept demanding too many permissions for their apps;  
 and cryptography often uses ECB mode as it’s the default with many crypto  
 libraries.

Customers and vendors want different things at multiple points in the value

chain. Regulation doesn’t always help, because governments have multiple agen-  
 das of their own, often in conﬂict: intelligence agencies, safety regulators and  
 competition authorities pull in different directions. It’s in this treacherous land-  
 scape that the assurance game is played.

Assurance is thus a political and economic process. It is also a dynamic

process, just like the development of code or of documents. Just as you have  
 bugs in your code, and in your speciﬁcation, you will also have things wrong  
 with your security and safety policies, leading to omissions and errors in your  
 test suite. So assurance is steadily turning from something done as a one-off  
 project to another aspect of continuous evolution.

With that warning, it’s helpful to start with the classic problem of evaluating

a static product that is built in a single project.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 913 | Ross Anderson |

*28.2. EVALUATION*

**28.2** **Evaluation**

Product evaluation tackles the problem of the lemons market we discussed in  
 section 8.3.3: when customers can’t measure quality, bad products drive out  
 good ones. Security has been a lemons market for generations. An 1853 book  
 on locksmithing justiﬁed disclosing the ‘secrets’ of the trade on the grounds  
 that the burglars knew them already; it was just the locksmiths’ customers  
 who were ignorant [1895]. Modern consumer-grade products, from anti-virus  
 software to mobile phone apps, are way beyond the ability of most consumers  
 to assess technically. If they are just going to rely on the brand name, the vendor  
 may as well buy ads rather than hiring security engineers. As for professional  
 products, the tech majors may employ enough PhDs to do an assessment, but  
 banks don’t – not even money-centre banks1. In earlier chapters, we discussed a  
 number of examples of static security standards against which various products  
 get evaluated and certiﬁed. Banks and governments are among the keenest

purchasers of certiﬁed security products.

That may have been where computer security got started ﬁfty years ago, but

as computers end up everywhere, we have to look at other industries too. Dozens  
 of industries have their own safety standards, with which security mechanisms  
 are increasingly intertwined. We already talked about electricity transmission  
 and distribution in section 23.8.1. Safety standards for software in road vehicles  
 have developed over decades; we talked about trucks in 14.3.3. Now that both  
 trucks and cars have multiple systems for assisted driving and are connected  
 to the Internet, they have critical security as well as safety requirements. The  
 same is happening for medical equipment and much else.

I’ll explore this via a number of case studies. Two important questions are

whether the evaluation is conducted by the relying party or by a third party,  
 and whether the standards are static or dynamic.

**28.2.1** **Alarms and locks**

The US insurance industry set up a joint testing lab in 1894, alarmed at the  
 ﬁre risks from electric lightbulbs; it was incorporated in 1901 as Underwriters’  
 Laboratories, a nonproﬁt that develops ﬁre safety and other standards, and  
 started approving security products in 1913 [1916]. Other countries have similar  
 bodies. An evaluator spends a ﬁxed budget of effort looking for ﬂaws and writes  
 a report, after which the lab either approves a device, turns it down or demands  
 some changes.

As the insurance industry bears much of the cost of ﬁres and burglaries,

incentives are somewhat aligned, although in practice these labs get much of  
 their income from testing fees. One risk is inertia: the standards may not keep  
 up with progress. In the case of high-security locks, a lab in 2000 might have  
 demanded ten minutes’ resistance to picking and say nothing about bumping.  
 We described in section 13.2.4 how bumping tools had improved enough to be

1In my late 20s and early 30s I worked in banking, and when I went to an interbank security

standards committee there were only about four of us in the room who knew what we were  
 talking about – of whom one was from IBM. Fintech has become an order of magnitude more  
 complex since then.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 914 | Ross Anderson |

*28.2. EVALUATION*

a major threat by 2010, and picks have got better too. We also described in  
 section 13.2.3 how bank vaults certiﬁed to resist attack for ten minutes can be  
 defeated in much less by a modern angle grinder or a burning bar. Insurance  
 labs in some countries, such as Germany, have been prepared to withdraw cer-  
 tiﬁcations as attacks got better; in the USA, they appear reluctant to, perhaps  
 for fear of being sued. The willingness of an industry to tolerate changing stan-  
 dards may depend on its structure: a mature industry with a handful of large  
 players can drag its feet a lot more than a growing competitive one.

**28.2.2** **Safety evaluation regimes**

Safety standards tend to emerge one industry at a time in response to major  
 accidents or scandals. The safety of drugs and medical devices is regulated in  
 the USA by the FDA, set up in 1906 by President Theodore Roosevelt after  
 journalists exposed abuses in the patent medicine industry. It turned out that  
 the top-selling medicine in America was just a dilute solution of sulphuric acid  
 and turpentine – really cheap to manufacture, yet tasting nasty enough that  
 people could believe it was good for them [2050]. As for air safety, the ﬁrst  
 step was in 1931, when America’s top football coach Knute Rockne died in a  
 plane crash caused by structural failure, causing a public outcry that led to the  
 establishment of the National Transportation Safety Board. The FAA was set  
 up later by President Eisenhower after a 1956 crash between two airliners over  
 the Grand Canyon killed all 128 people aboard the two planes [684]. As for the  
 car industry, it managed to disclaim liability for safety for decades. Vendors  
 competed to decorate cars with chromium rather than ﬁt them with seat belts,  
 until Ralph Nader’s book ‘Unsafe at Any Speed’ spurred Congress to set up  
 National Highway Traffic Safety Administration (NHTSA) in 1970; its power  
 and inﬂuence grew with successive safety scandals.

Europe harmonised a patchwork of national laws into the Product Liability

Directive in 1985, adding further regulations and safety agencies by industry  
 sector. Since then, the European Union has developed into the world’s lead  
 safety regulator, with its agencies setting safety standards in industries from  
 aviation through railway signals to toys [1148]. With cars, for example, Europe  
 generally requires safety testing by independent labs2, while America doesn’t;  
 but most US vendors have their US models tested independently too, as Europe  
 created the ‘industry norm’ by which US courts assess tort cases when things  
 go wrong. In this sense, Europe has become a ‘regulatory superpower’.

The EU’s overall safety strategy is to evolve a set of standards by negotiation

with industry working groups and lobbyists and update them every seven to  
 ten years. Many products that cause serious harm, such as cars, have to get  
 explicit approval, typically following testing in an independent laboratory. Less  
 dangerous goods such as toys require self-certiﬁcation: the vendor places a ‘CE’  
 mark on the product to assert that it complies with all relevant standards.  
 This removes some of the excuses that vendors might use when non-compliant

|  |  |  |
| --- | --- | --- |
| 2Europe delegates type approval to Member States, most of which have a Type Approval  Authority which delegates testing to a specialist lab. In Germany, that’s T¨UV. Some smaller | | |
| countries have a TAA that allows the manufacturer to do its own testing, with a TAA inspector present. | | |
| **Security Engineering** | 915 | Ross Anderson |

*28.2. EVALUATION*

products cause accidents; it’s also used for a wide range of components from car  
 brakes to industrial pressure valves.

**28.2.3** **Medical device safety**

Safety regulation is a complex ecosystem, imperfect in many ways. For example,  
 there has long been controversy in both America and Europe over medical device  
 safety. This came to prominence in the 1980s when bugs in the Therac 25

medical accelerator caused the death of three patients and injured three more.  
 The cause was a software bug that surfaced as a usability issue: if the operator  
 edited the machine’s parameters too quickly, they could get the machine into a  
 dangerous state where it delivered far too much radiation to the patient. The  
 case study is set reading for my software engineering students even today [1149].

Figure 28.1: – two infusion pumps that are apparently of the same model (photo  
 courtesy of Harold Thimbleby)

The most lethal medical devices nowadays are probably infusion pumps, used

to administer intravenous drugs and other ﬂuids to patients in hospital. Many  
 of the fatal accidents are usability failures. Just look at Figure 28.1: each of  
 these claims to be a ‘BodyGuard 545’ yet to increase the dose on the machine  
 on the left, you press ‘2’ while on the right you press ‘5’. An emergency room  
 might have equipment from half-a-dozen different vendors, all with different user  
 interfaces. Doctors and nurses occasionally press the wrong button, the wrong  
 dose gets administered, or the dose for an eight-hour transfusion is given all in  
 one bolus – and patients die. Infusion pumps kill about as many people as cars  
 do, with the body count being in the low thousands in the UK and the low tens  
 of thousands in the USA [1878].

Surely this could be ﬁxed with standards? Well, there are standards. For

example, ‘litres’ is supposed to be marked with a capital ‘L’ so it’s not mistaken  
 for a ‘1’, but you can see on the right-hand image that although the ‘0L/h’  
 complies with this, the ‘500ml’ does not. So why is the standard not enforced?  
 Well, the FDA budget of engineering effort is about half a day per device,  
 and vendors don’t give the engineers actual devices to play with. It’s just a  
 paperwork review3. In addition, usability falls outside the FDA’s scope. This

3By way of comparison, when colleagues and I helped to evaluate a burglar alarm designed

for low-consequence risks such as small shops and houses, our budget was two person-weeks.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 916 | Ross Anderson |

*28.2. EVALUATION*

is, I hear, a result of lobbying by the industry to ‘cut red tape’. The fact that  
 two different devices are marketed as the same product is a common strategy  
 to minimise compliance costs.

There has recently been international guidance for usability engineering of

medical devices in the form of ISO/IEC 62366-2, which took effect in 2018.  
 This is a signiﬁcant advance which covers a lot of ground, but usability is a huge  
 ﬁeld. The new standard is very basic, and explains at length that manufacturers  
 should not just list hazards in a legal warning leaﬂet, or even highlight them  
 with notices on the equipment – they should actually try to mitigate them, and  
 in the process understand how their equipment is likely to be used and abused.  
 It describes a number of assessment techniques the engineer could use, but  
 “insufficient experience with the type of medical device” is just one bullet point  
 on its list of factors that might contribute to use errors. Manufacturers will ﬁnd  
 all this expensive, and will no doubt talk to their lawyers about how much really  
 has to be done. Safety in number entry alone is a complex ﬁeld [1879]; every  
 vendor should probably train an expert in it, and in dozens of other techniques  
 too, but many will do as little as they think they can get away with. In the  
 end, a usability assessment will now be in the trolleyload of paperwork the  
 manufacturer presents to regulators, at least outside the USA. But it’s unclear  
 whether the confusion arising when nurses also use the different interfaces of  
 competitors’ equipment will be taken as seriously as it should be.

This is all teaching us that pre-market testing isn’t enough for medical de-

vice safety – you need diligent post-market surveillance too. This started to  
 be introduced throughout Europe in 2017 following a scandal about defective  
 breast implants [233]. In the UK, a further scandal about teratogenic drugs  
 and pelvic mesh implants let to an Independent Medicines and Medical De-  
 vices Safety Review, which in 2020 documented decades of indifference to safety  
 and recommended among many other things that regulation ‘needs substantial  
 revision particularly in relation to adverse event reporting and medical device  
 regulation’ [503]. In May 2020, a new EU medical device regulation (2017/745)  
 was supposed to require post-market surveillance systems and a public database  
 of anonymised incident reports; implementation was postponed until May 2021.  
 And in June 2020, the UK Parliament passed a Medicines and Medical Devices  
 Act that will enable ministers to amend the existing regulations after Brexit.  
 The mood music there, however, is to make Britain a more attractive place  
 for drug companies and medical device makers, not a safer place for patients.  
 Within Britain’s National Health Service, it’s hard to make a career as a safety  
 specialist4.

Now here’s an interesting question. If infusion pumps kill as many people

as cars or – in the USA – as guns, why aren’t people more worked up, as they  
 are about road safety and gun control? Well, the harm is both low-key and  
 diffuse. At your local hospital, such accidents probably kill less than one person  
 a month, and many of them won’t be noticed, as people on infusion pumps tend  
 to be fairly sick anyway. When they are noticed, they are more likely to be  
 blamed on the nurse, rather than on the medical director who bought pumps

4The UK NHS has a Healthcare Safety Investigations Branch, established in 2016, but it

investigates what it’s told to, often has to keep its ﬁndings conﬁdential, and doesn’t have or  
 seek enforcement powers to require other healthcare organisations to make changes [875].

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 917 | Ross Anderson |

*28.2. EVALUATION*

from half a dozen different suppliers following nice lunches with the sales folks.  
 As a cause of death in the hospital, recorded safety usability failures don’t make  
 it into the top twenty, and so don’t get attention from politicians or the press.  
 (The exception is when a safety failure has a security angle, as people are very  
 sensitive indeed about hostile intent. I’ll discuss this in section 28.4.2 below.)

The standardisation of user interfaces is managed better in industries where

accidents and their causes are more visible. Road traffic accidents are fairly  
 visible and most people drive, so car crashes and their causes are a topic of  
 conversation. The controls in cars are now fairly standard, with the accelerator  
 on the right, the brake in the middle and the clutch on the left. Things aren’t  
 perfect; if you’re in a hurry, you might get in a rental car, drive off down the  
 freeway, then struggle to ﬁnd the light switch as night falls. But it used to be  
 much worse. Some cars in the 1930s had the accelerator in the middle, while  
 the ﬁrst mass-produced car, the Model T Ford, had a hand throttle and a pedal  
 gear-change, like a motorcycle. The average modern driver would have a hard  
 time getting such a car out of the rental lot.

**28.2.4** **Aviation safety**

Aviation has much stronger safety incentives still: airliners are worth eight or  
 nine ﬁgures, crashes are front-page news, they cause pilots as well as passengers  
 to lose their lives and airline CEOs may even lose their bonuses. Pilots pay  
 attention to accident reports, and are required to train on each type of plane  
 they ﬂy. This has led the vendors to standardise cockpit design, starting with  
 the Boeing 757 and 767, which were designed from the start to be so similar that  
 a pilot trained on one could ﬂy the other. If nurses were similarly required to  
 get a type rating for each infusion pump, that would cost real money, hospital  
 executives would pay attention, the vendors would eventually follow Boeing,  
 and a lot of lives could be saved.

Yet we ﬁnd regulatory failure in aviation too, and an example was exposed

with the Boeing 737Max crashes. Since Boeing had bought McDonnell Dou-  
 glas in 1997 and become the only US ﬁrm making large aircraft, the Federal  
 Aviation Administration had come to see its role as supporting Boeing. The  
 company’s engineers were allowed to take over much of the safety evaluation  
 and certiﬁcation work that the FAA had done in the past. An even more toxic  
 effect of the takeover was that McDonnell Douglas executives took over, the  
 company moved its headquarters from Seattle to Chicago, and was no longer  
 run by engineers but by ﬁnance people who had already destroyed one engi-  
 neering company and whose goal now was to milk the maximum proﬁts from  
 the new monopoly. Boeing’s traditional engineering culture was sidelined and  
 corners were cut [729]. Two crashes followed, in Indonesia and Ethiopia, killing  
 346 people. The cause was reminiscent of the Therac case a generation earlier:  
 a design error in software that surfaced as a life-threatening usability failure.

In order to compete with the latest model of Airbus, Boeing needed to make

the 737 more fuel efficient quickly, and this meant larger engines, which had to be  
 ﬁtted further forward, or it would have required re-engineering the airframe to  
 the point that it would have been a new plane for regulatory purposes, and would  
 have taken much longer to certify. The new engine location made the aircraft

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 918 | Ross Anderson |

*28.2. EVALUATION*

harder to trim at high speeds, so Boeing added software called the Maneuvering  
 Characteristics Augmentation System (MCAS) to the ﬂight control computer  
 to compensate for this.

The MCAS software needed to know the aircraft’s angle of attack, and the

critical design error was to rely on one angle-of-attack sensor rather than two,  
 although these are often damaged by ground handlers and bird strikes. The  
 implementation error was that, with an incorrect angle-of-attack input, the plane  
 could get into a regime where the pilots needed to pull about 50kg on the yoke  
 to keep the plane level. This was compounded by an error in safety analysis:  
 the unintended activation of the MCAS software was not anticipated. As a

result, Boeing didn’t do a proper failure modes and effects analysis and the  
 software’s behaviour was not even documented in the pilot manual. The pilots  
 were not trained how to diagnose the problem or switch MCAS off. Boeing  
 had become complacent about the ability of pilots to cope with the chaos of a  
 cockpit emergency with many alarms going off at once [1055].

The company had also got away with bullying investigators over a similar

previous crash in the Netherlands in 2009, and initially hoped that the Indonesia  
 crash could be blamed on pilot error [857]. The FAA responded to the crash  
 by sending an emergency airworthiness directive to all known U.S. operators of  
 the airplane, which consisted of inserting a warning notice in the airplane ﬂight  
 manual [665]. ; However, the warning light that alerted pilots to disagreement  
 between the two sensors had been made an airline option, like a sun roof in a  
 car, and the operation of the switch that could disable MCAS was changed to  
 make it less intuitive [155]. A number of U.S. pilots logged complaints, with one  
 describing the manual as ‘almost criminally insufficient’ [139]; but the FAA saw  
 such complaints as only relevant to air carrier operations and did not analyse  
 them for global safety hazards [664].

After the second crash in Ethiopia, other countries’ regulators started ground-

ing the 737Max, and the FAA could no longer protect them. Boeing had lost  
 $18.7bn in sales by March 2020, when the coronavirus pandemic closed down  
 commercial aviation sales, as well as $60bn in market capitalisation. This was  
 by some distance the world’s biggest ever software failure, in terms of both lives  
 lost and economic damage. The ﬁx, approved in August 2020, involves not just  
 a software change so that MCAS reads both angle-of-attack sensors and deploys  
 only once per ﬂight and with limited stick force; but a procedural change so  
 that both sensors are checked pre-ﬂight; an update to pilot training; and a reg-  
 ulatory change so that the FAA, rather than Boeing, checks each plane after  
 manufacture [592].

When analysing safety, it’s not enough to think of it as a technical test-

ing matter. Psychology, incentives, institutions and power matter too. The

power of lobbyists, and the risk that regulators will be captured by the in-  
 dustry they’re supposed to regulate, place real limits on what can be achieved  
 by testing regimes. Over time, measures designed for risk assessment and risk  
 reduction become industrialised and tend to become a matter of compliance,  
 which ﬁrms then seek to pass at minimum cost. It’s also important to stop  
 thinking of problems as ‘aerospace engineering’ versus ‘software engineering’,  
 or ‘safety engineering’ versus ‘security engineering’. If you want to be a good  
 engineer you need to try to understand every aspect of the whole system that

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 919 | Ross Anderson |

*28.2. EVALUATION*

might be relevant.

**28.2.5** **The Orange Book**

The ﬁrst serious computer security testing regime was the *Orange Book* – the  
 Trusted Computer Systems Evaluation Criteria [544]. We touched on this in  
 section 9.4, where I described the multilevel security model that the US Depart-  
 ment of Defense was trying to promote through it. Orange Book evaluations  
 were done from 1985–2000 at the NSA on computer systems proposed for govern-  
 ment use and on security products such as cryptographic devices. In incentive  
 terms, it was a collective relying-party scheme, as with insurance.

The Orange Book and its supporting documents set out a number of eval-

uation classes, in three bands. C1 meant just that there was an access-control  
 system; C2 corresponded to carefully conﬁgured commercial systems. In the  
 next band, B1 meant mandatory access control; B2 added covert channel anal-  
 ysis, a trusted path to the TCB from the user, and severe penetration testing;  
 while B3 required the TCB had to be minimal, tamper-resistant, and subject  
 to formal analysis and testing. At the top band, A1 added a requirement for  
 formal veriﬁcation. (Very few systems made it to that level.)

The evaluation class of a system determined what spread of information

could be processed on it. The example I gave in section 9.6.2 was that a system  
 evaluated to B3 could process information at Unclassiﬁed, Conﬁdential and  
 Secret, or at Conﬁdential, Secret and Top Secret.

When the Orange Book was written, the Department of Defense thought that

they paid high prices for high-assurance computers because the markets were  
 too small, and hoped that security standards would expand the market. But  
 Orange Book evaluations followed government work practices. A government  
 user would want some product evaluated; the NSA would allocate people to do  
 it; given traditional civil service caution and delay, this could take two or three  
 years; the product, if successful, would join the evaluated products list; and the  
 bill was picked up by the taxpayer. Evaluated products were always obsolete,  
 so the market stayed small, and prices stayed high5.

Other governments had similar ideas. European countries developed the

*Information Technology Security Evaluation Criteria* (ITSEC), a shared scheme  
 to help their defense contractors compete against US suppliers. This introduced  
 a pernicious innovation – that the evaluation was not arranged by the relying  
 party (the government) but by the vendor. Vendors started to shop around  
 for the lab that would give their product the easiest ride, whether by asking  
 fewer questions, charging less money, taking the least time, or all of the above.  
 Contractors could obtain approval as a *commercial licensed evaluation facility*  
 (CLEF), and in theory the CLEF might have its license withdrawn if it cut  
 corners. That never happened.

5To this day, most governments are hopeless at buying technology and pay several times

the market rate, if they make it work at all. The reasons are much broader and deeper than  
 standards. See for example section 10.4.4 on the £11bn failure of a project to modernise

Britain’s National Health Service, and section 23.5 for the $6bn failure of the Pentagon’s  
 Joint Tactical Radio System.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 920 | Ross Anderson |

*28.2. EVALUATION*

**28.2.6** **FIPS 140 and HSMs**

The second evaluation scheme promoted by the US government in the 20th  
 century was NIST’s FIPS 140 scheme for assessing the tamper-resistance of  
 cryptographic processors. This was aimed at helping the banking industry as  
 well as the government, and as I described in section 18.4 it uses a number  
 of independent laboratories as contractors. Launched in 1994, it is still going  
 strong today, and is favoured by US customers of cryptographic equipment.

There are two main failure modes of FIPS 140. The ﬁrst is that it covers the

cryptographic device’s hardware, not its software, and many FIPS 140 evaluated  
 devices (even at the highest levels) run applications with intrinsic vulnerabil-  
 ities. Weak algorithms, legacy modes of operation and vulnerable APIs are

mandated by bank standards bodies for backwards compatibility, as described  
 in section 20.5. The ﬁx for this has been a growing emphasis on standards

set by PCI, the payment industry’s self-regulation scheme, which I describe in  
 section 12.5.2.

The second is that the FIPS 140-1 standard has a big gap between level 3

and level 4 for historical reasons I discussed in section 18.4. FIPS 140 level 3 is  
 easy to obtain (you just pot the circuit in epoxy to make it inaccessible to casual  
 probing) and some level-3 devices are not too hard to break (you just scrape off  
 the epoxy with a knife). Level 4 is really hard, and only a few devices ever made  
 that grade. So many vendors aim at what the industry calls, informally, ‘level  
 3.5’. As this doesn’t have any formal expression in the FIPS standard, ﬁrms  
 often rely on the Common Criteria instead when talking to customers outside  
 the USA.

**28.2.7** **The Common Criteria**

This sets the stage for the Common Criteria. Following the collapse of the

Soviet Union in 1989, military budgets were cut, and it wasn’t clear where the  
 opponents of the future would come from. Eventually the US and its allies

agreed to scrap their national schemes and replace them with a single standard  
 – the Common Criteria for Information Technology Security Evaluation [1396].

The work was substantially done in 1994–1995, and the European ITSEC

model won out over the Orange Book approach. Evaluations at all but the high-  
 est levels are done by CLEFs, are supposed to be recognised in all participating  
 countries, and vendors pay for them.

The innovation was support for multiple security policies. Rather than ex-

pecting all systems to conform to Bell-LaPadula, the Common Criteria evaluate  
 a product against a *protection proﬁle* (PP), which is a set of security functional  
 requirements and assurance requirements for a class of product. You can think  
 of it as a detailed security policy, but oriented at products rather than systems,  
 and expanded into several dozen pages of detail. There are protection proﬁles for  
 operating systems, access control systems, boundary control devices, intrusion  
 detection systems, smartcards, key management systems, VPN clients, voting  
 machines, and even transponders that identify when a domestic waste bin was  
 last emptied. Anyone could propose a protection proﬁle and have it evaluated

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 921 | Ross Anderson |

*28.2. EVALUATION*

by the lab of their choice. It’s not that the defence community abandoned mul-  
 tilevel security, so much as tried to mainstream its own evaluation system by  
 getting commercial ﬁrms to use it for other purposes too. But an evaluation  
 depends entirely on what was measured and how. Some aspects of security were  
 explicitly excluded, including cryptography, emission security (as the NATO  
 standards were classiﬁed) and administrative procedures (which was bad news  
 for usability testing).

The Common Criteria have enjoyed some limited success. Its evaluations

are used in specialised markets, such as smartcards, hardware security modules,  
 TPMs and electronic signature devices, where sectoral due-diligence rules (such  
 as PCI) or regulation (such as electronic signature laws) create a compliance  
 requirement. Evaluations of such devices were kept honest for a while by an  
 informal cartel run by SOG-IS (the senior officials group – information security)  
 – a committee of representatives of the intelligence agencies of EU countries.  
 However the operation of the CC outside Europe has been a bit of a joke, and  
 even within Europe it has been undermined by both companies and countries  
 gaming the system. The UK withdrew in 2019.

**28.2.7.1** **The gory details**

To discuss the Common Criteria in detail, we need some jargon. The product  
 under test is known as the *target of evaluation* (TOE). The rigor with which the  
 examination is carried out is the *evaluation assurance level* (EAL) and can range  
 from EAL1, for which functional testing is sufficient, all the way up to EAL7  
 which demands not only thorough testing but a formally veriﬁed design. The  
 highest evaluation level commonly obtained for commercial products is EAL4,  
 although in 2020 there are 85 products at EAL6 or above out of 1472 certiﬁed  
 under CC, and many smartcards are evaluated to EAL4+ which means EAL4  
 plus one or more of the requirements set at higher levels.

When devising something from scratch, the idea is to ﬁrst work out a threat

model, then create a security policy, reﬁne it to a *protection proﬁle* (PP) and  
 evaluate it (if a suitable one doesn’t exist already), then do the same for the  
 security target, then ﬁnally evaluate the actual product. A protection proﬁle  
 consists of security requirements, their rationale, and an EAL, all for a class  
 of products. It’s supposed to be expressed in an implementation-independent  
 way to enable comparable evaluations across products and versions. A *security*  
 *target* (ST) is a reﬁnement of a protection proﬁle for a speciﬁc product. One  
 can evaluate a PP to ensure that it’s complete, consistent and technically sound,  
 and an ST too. The evaluations are ﬁled with the national authority, which is  
 typically the defensive arm of the local signals intelligence agency. The end  
 result is a registry of protection proﬁles and a catalogue of certiﬁed products.

There is a stylized way of writing a PP or ST. For example, FCO\_NRO is a

functionality component (hence F) relating to communications (CO) and it refers  
 to non-repudiation of origin (NRO). Other classes include FAU (audit) and FCS  
 (crypto support).

There are also catalogues of

*• threats*, such as T.Load\_Mal – “Data loading malfunction: an attacker

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 922 | Ross Anderson |

*28.2. EVALUATION*

may maliciously generate errors in set-up data to compromise the security  
 functions of the TOE”

*• assumptions*, such as A.Role\_Man – “Role management: management of  
 developers, operators and so on behave themselves)

*• organizational policies*, such as P.Crypt\_Std – “Cryptographic standards:  
 be in accordance with ISO and associated industry or organizational stan-  
 dards”

*• objectives*, such as O.Flt\_Ins – “Fault insertion: the TOE must be resis-

*• assurance requirements*, such as ADO\_DEL.2 – “Detection of modiﬁcation:  
 of it to the user”

A protection proﬁle should now contain a *rationale*, which typically consists

of tables showing how each threat is controlled by one or more objectives, and  
 in the reverse direction how each objective is necessitated by some combination  
 of threats and environmental assumptions. It will also justify the selection of  
 an assurance level and requirements for strength of mechanism.

The fastest way to get the hang of this may be to read the core CC docu-

mentation itself, then a few proﬁles. The quality varies widely. For example, a  
 protection proﬁle for automatic cash dispensers, written in management-speak  
 with clip art, ‘has elected not to include any security policy’ and misses many of  
 the problems that were well known when it was written in 1999 [340]. A proﬁle  
 for voting machines from 2007 [563] was written more in politicians’ language,  
 but at least with reasonable clarity6.

Protection proﬁles for smartcards emphasise maintaining conﬁdentiality of

the chip design by imposing NDAs on contractors, shredding waste and so  
 on [650], while in practice most attacks on smartcards used probing or power-  
 analysis attacks for which knowledge of the chip mask was irrelevant. This has  
 developed into a political row, as I discussed in section 18.6.4: the smartcard  
 vendors have pushed the evaluation labs into demanding that all cryptographic  
 products be secure against ‘advanced persistent threats’. The ﬁght is over as-  
 surance requirement AVA\_VAN.5 which essentially requires that the entire de-  
 velopment environment should be air-gapped, like the Top Secret systems at  
 an intelligence agency. An air gap in itself won’t stop a capable opponent, as  
 the Iranians found out with Stuxnet and the Americans with Snowden; but it  
 causes real inconvenience to normal IT companies who rely on Github and other  
 cloud-based systems. And that’s entirely the point: the smartcard ﬁrms don’t  
 want HSMs or enclaves encroaching on their markets.

6This appears designed to support French ﬁrms’ drive to export population registration

systems, and it is these rather than the actual voting machines that are often the real weak  
 point in elections – as I discussed in section 7.4.2.2.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 923 | Ross Anderson |

*28.2. EVALUATION*

**28.2.7.2** **What goes wrong with the Common Criteria**

By the time the second edition of this book came out in 2008, industry people  
 had a lot of complaints about the Common Criteria, which I discussed there  
 and which I update more brieﬂy here.

*•* The biggest complaint for years has been the cost and bureaucracy of the  
 have to spend several million Euros and several years of effort to navi-  
 gate the process. In practice the CC have become a moat that defends  
 established cartels.

*•* The next biggest is that, as well as avoiding ‘technical physical’ aspects  
 measures, which means in practice ignoring usability. In general, user

interfaces are considered to be somebody else’s problem.

*•* Protection proﬁles are designed by their sponsor ﬁrms to rig the market.  
 also use air-gapped systems to push their costs up. The gaming often

leads to insecure products: vendors write their PPs to cover the things  
 they can do easily. They might evaluate the boot code, but leave most of  
 the operating system outside the scope. Recall the API attacks on HSMs  
 described in section 20.5; some vulnerable HSMs were CC-certiﬁed, and  
 similar failures are seen in other CC-certiﬁed products too.

*•* Sometimes the protection proﬁles might be sound, but the way they’re  
 pean eIDAS regulation which requires businesses to recognise digital sig-  
 natures made using smartcards, and encouraged governments to demand  
 them for interactions such as ﬁling tax returns. The main problem in this  
 application, as I discussed in section 18.6.1, is the lack of a trusted inter-  
 face. As that problem’s too hard, it’s excluded, and the end result is a  
 ‘secure’ signature on whatever the virus or Trojan in your PC sent to your  
 smartcard. This hole was duly slathered with several layers of fudge. PPs  
 were written for a smartcard to function as a ‘Secure Signature-Creation  
 Device’; other PPs appeared for HSMs, and for the *signature activation*  
 *module* (SAM) – the server software that passes them digital objects to  
 be signed. The HSM plus the SAM are evaluated as a *qualiﬁed signa-*  
 *ture creation device* (QSCD) [29]. But the front-end server software used  
 by the service provider is only audited, not certiﬁed, and if you’re lucky  
 the app on your phone or tablet might have RASP on it as a malware  
 countermeasure, as I discussed in section 12.7.4. That is what lobbyists  
 can achieve: the whole certiﬁcation machinery has been twisted to allow  
 services like Docusign inside the tent, so long as they use a CC certiﬁed  
 HSM to hold their signature keys.

*•* The CC claim not to assume any speciﬁc development methodology, but  
 of policy evolving in response to experience but re-evaluation of PPs or  
 products is declared to be outside the scope. So they’re unable to cope with

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 924 | Ross Anderson |

*28.2. EVALUATION*

normal security development lifecycles, or with commercial products that  
 get monthly security patches. (The same goes for FIPS; of the available  
 standards, only PCI can cope with updates.)

*•* The Criteria are technology-driven, when in most applications it’s the  
 the hard way that hand-marked paper ballots are way better than voting  
 machines for all sorts of reasons. Security is a property of systems, not of  
 products.

*•* The rigour of the evaluations varies widely between countries, with Ger-  
 lands in the middle, while Spain and Hungary let their CLEFs give spon-  
 sors an easy ride. Nobody within the system can actually say this in public  
 without causing a diplomatic incident, so it cannot be ﬁxed. The costs  
 also vary, with an evaluation in Germany costing perhaps three times what  
 you pay in Hungary.

*•* The Common Criteria brand isn’t well defended. I described in sec-  
 tion 12.6.1.1 how PIN entry devices claimed by VISA to have been eval-  
 uated under the Common Criteria were insecure; GCHQ’s response was  
 that as the evaluation had not been registered with them, and the devices  
 were not claimed to be ‘CC certiﬁed’ it wasn’t their problem. So suppliers  
 are free to continue describing a defective terminal as ’CC evaluated’. A  
 business would not tolerate such abuse of its trademark.

*•* More generally, there’s nothing on liability: ‘The procedures for use of

In the second edition of this book, I took the view that Common Criteria

evaluations were somewhat like a rubber crutch. Such a device has all sorts  
 of uses, from winning a judge’s sympathy through wheedling money out of a  
 gullible government to whacking people round the head. Just don’t try to put  
 serious weight on it.

**28.2.7.3** **Collaborative protection proﬁles**

In an attempt to deal with these criticisms, collaborative protection proﬁles  
 (cPPs) started to appear in 2015. The idea was to move away from the EAL  
 levels towards a single protection proﬁle for each class of secure device, and to  
 develop that proﬁle as a collaborative effort among ﬁrms in an industry, with  
 input from government and academics [462]. The hope was to stop security  
 evaluations being abused in strategic games between competitor ﬁrms. The

results of this can now be seen in 2020 by browsing the catalogue of evaluated  
 products on the CC website. Vendors in France and Germany still offer many  
 smartcards, and related products such as electronic signature creation devices,  
 with certiﬁcates at EAL4+ or EAL6; that’s the legacy of the SOG-IS cartel.

Outside Europe, though, the CC system has been completely captured by

vendor interests. American ﬁrms offer many ﬁrewalls, routers and other net-  
 working products, evaluated according to industry cPPs; and Japanese ﬁrms

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 925 | Ross Anderson |

*28.2. EVALUATION*

offer a range of printers and fax machines. So what is a secure fax machine  
 – does it encrypt faxes? Not at all; it just behaves as you’d expect a fax ma-  
 chine to (if you’re old enough to remember them). In short, cPPs have become a  
 marketing mechanism, and are now undermining the traditional CC core. Firms  
 wanting to sell electronic signature systems can have them evaluated under a  
 cPP which is considered EAL4, and most customers can’t tell the difference  
 between that and an EAL4+ evaluation done under the old rules.

**28.2.8** **The ‘Principle of Maximum Complacency’**

There’s a substantial literature on the economics of standards, as there are  
 many contexts in which people have to choose between them. If you’re a bright  
 teenager, do you apply to a top university and risk getting a second-class degree,  
 or should you go to a local college and be a star? Should you worry about grade  
 inﬂation eroding the value of your degree in either case? If you’re raising money  
 for a startup, should you get your money from business angels or try to get a  
 big-name venture fund on board? An IT vendor wondering whether to go for  
 some kind of certiﬁcation faces somewhat similar choices. And even nations  
 play certiﬁcation games. The large service ﬁrms all have their EU headquarters  
 in Ireland as it has long been Dublin’s policy to have the most relaxed regime  
 of privacy regulation in Europe, as well as the lowest corporate taxes. What  
 options are there for dealing with such games?

The most inﬂuential model of such choices is a 2006 paper on forum shop-

ping by Josh Lerner and Jean Tirole7. Their model is a three-stage game in  
 which the sponsor selects a certiﬁer, the certiﬁer then studies the offering and  
 perhaps demands some changes, and ﬁnally the end-users make decisions to  
 buy or not [1143]. The big question is whether competition between certiﬁers  
 will result in better standards, or in a race to the bottom. In most cases the  
 *principle of maximum complacency* wins out: owners seek endorsement from a  
 single certiﬁer, and resist attempts to get them to improve the product. Only in  
 certain circumstances can competition improve quality. One example is where  
 NGOs compete to certify products as sustainable: there, the certiﬁer cares more  
 about the users’ outcome than the sponsors do, and the desired property isn’t  
 strongly controlled by a single sponsor. Another is competition between elite  
 universities: students have no market power, and enough employers will pay a  
 premium for elite graduates that there’s plenty of incentive for Cambridge to  
 compete with Oxford, MIT and Berkeley.

Where there are more players than just the sponsor, the certiﬁer and the

users, things get more complicated.

Certiﬁcation games take place in a much larger ecosystem. A company in-

vents some new product and sells it to some customers. The customers then  
 want a standard, and some tests to satisfy their auditors. They may want the  
 inventor to license the product to their established suppliers, or at least to a  
 second supplier. Other inventors pile in, and all of a sudden there’s a patent  
 pool. The ﬁrms negotiate long and hard to get their patents in to maximise  
 their share of the royalties; this often results in horrible standards that are in-

7Tirole won the 2014 Nobel for this and much other work in market power and regulation.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 926 | Ross Anderson |

*28.2. EVALUATION*

secure and hard to ﬁx (see section 14.2.4 on smart meters; there are many more  
 examples). The patent pools may become cartels that prevent new market en-  
 trants; this complaint has been made of the GSMA standards around 5G (see  
 section 22.2.4). The GSMA has also been criticised for its *Network Equipment*  
 *Security Assurance Scheme* (NESAS) where the vendor pays for a security as-  
 sessment that only takes a few days (and now allows remote audits because of  
 the pandemic). In short, industrial strategy doesn’t optimise for great products  
 so much as for monopolies or cartels.

Where a market is dominated by a monopoly, customer and political pressure

may eventually cause the monopolist to pay attention to security, and it can even  
 be rational for a monopolist to internalise some of the security externalities  
 (see the Microsoft case in section 27.5.3). But in the general case, of complex  
 supply chains with some steps dominated by cartels, it can be a lot harder.  
 The complexities in security certiﬁcation are roughly (a) the relying parties –  
 those at risk if the thing gets hacked – may be customers, third parties such  
 as insurers, or the public (b) the sponsors may be vendors, customers, relying  
 parties or associations of any of these (c) the testers may compete on price  
 or on quality, and this means the lowest quality threshold they can get away  
 with subject to not losing a license from an accreditation body, which may be  
 a government entity or a trade association (d) there may be more than one  
 accreditation body, plus politics between them. So we can have multiple layers  
 of indirection and we occasionally even get competition about “who certiﬁes the  
 certiﬁers”. To make sense of things we have to look at actual cases in detail.

In the case of CC evaluated products at EAL4 or above such as smartcards

and HSMs, suppose Alice’s company sells a product to Bob’s Bank and gets  
 Charlie the certiﬁer to say it’s secure, after which Bob’s customer Dorothy  
 defrauds another customer Eve and absconds. How does the evaluation change  
 things when Eve now claims her money back from Bob in court? Bob will

argue he wasn’t negligent because he operated according to the standards of the  
 industry, so isn’t liable to reimburse Eve. This argument is even more powerful  
 if Charlie signed off on his system. Charlie’s role is not so much a technical  
 authority as a liability shield. So Alice will work only as hard as she has to  
 to satisfy Charlie. Charlie will compete with his competitors and a race to the  
 bottom will ensue. The upshot in real life was that the payment card brands  
 set up PCI to take over Charlie’s role. We discussed in section 12.5.2 how

such standards shift liability in banking: they protect the bank more than the  
 merchant (surprise, surprise).

In the case of electronic signature devices, as we discussed in section 28.2.7.2

above, smartcard industry lobbying led Europe to pass signature laws that gives  
 special force to signatures created with certiﬁed products, even when these are  
 insecure. Lobbying by online service signature providers such as Docusign got  
 them on board too. The ultimate effect is not security but a tax. (And to ﬁle  
 a tax return in some EU countries you have to get it signed by such a service,  
 adding an extra twenty Euros to your tax accountant’s fee.)

So should certiﬁcation be voluntary? An interesting case study is by Ben

Edelman of the Trust-e scheme to certify websites. He discovered that certiﬁed  
 websites were more likely to attempt to load malware on to your computer,  
 rather than less. Adverse selection turned the scheme into a negative signal of

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 927 | Ross Anderson |

*28.2. EVALUATION*

quality: the weaker vendors certiﬁed their websites, while well-known consumer  
 brands didn’t bother [612]. The reason for this was that Trust-e certiﬁcation,  
 being voluntary, was cheap, and the technical barrier to certiﬁcation was also  
 low.

But although industry lobbies like to talk of ‘cutting red tape’, how many

might be happy with the outright abolition of a government-backed safety or  
 security standard or agency? In practice, lobbyists seek to capture regulators  
 rather than abolish them. Many regulatory regimes function both as moats to  
 prevent incumbents being challenged too easily by startups and also as liability  
 shields. As an example, we discussed in section 17.3 how Amazon, Microsoft,  
 Google and IBM have restricted sales of face-recognition software – among the  
 most controversial of their products – until it’s regulated.

**28.2.9** **Next steps**

Since Brexit, the UK and Europe have diverged. Europe passed a Cybersecu-  
 rity Act (regulation 2019/881) which strengthens the European Network and  
 Information Security Agency (ENISA) and places it at the centre of its strat-  
 egy. ENISA is to act as a centre of expertise and liaise with sectoral regulators  
 in banking, aviation, energy and telecomms, as well as the data protection au-  
 thorities. I expect this will be of major importance in the long run, as safety  
 and security regulation are coming together and will inevitably be managed on a  
 sectoral basis by the standards bodies for cars, aircraft, medical devices, railway  
 signals and so on. I will return to this later.

As for the certiﬁcation of information security products, its approach might

be described as ‘one more heave’: it is setting up an *EU Cybersecurity Cer-*  
 *tiﬁcation Framework* under ENISA which will take over as the top-level certi-  
 ﬁer. It’s supposed to “help avoid the multiplication of conﬂicting or overlapping  
 national cybersecurity certiﬁcation schemes and thus reduce costs for under-  
 takings operating in the digital single market” [655]. It will apply to services  
 and processes as well as products. As I write in 2020, the details are still

being worked out, but the intention is that sponsoring bodies of EU member  
 states will run certiﬁcation at three levels, ranging from ‘basic’ which entails the  
 vendor self-assessing conformance with standards and assuming responsibility  
 for compliance, through ‘substantial’ which will involve veriﬁcation of security  
 functionality, to ‘high’ which will involve ENISA taking over from SOG-IS the  
 supervision of the smartcard / HSM / e-signature kit currently evaluated at  
 EAL4 and above.

The UK government was concerned about certiﬁcation for many years and

was involved in pushing cPPs in order to try to make certiﬁcation more stan-  
 dardised. But by 2017 they had come to the conclusion that the Criteria were  
 neither necessary nor sufficient for security, and GCHQ withdrew as a sponsor  
 from 2019. It no longer licenses CLEFs or approves certiﬁcations, although UK  
 organisations may continue to use certiﬁcations created elsewhere8. It has long  
 had its own national product certiﬁcation scheme, now known as *commercial*

8One of my spies in the Doughnut says ‘We absolutely recognise any CC certiﬁcate from

any producing nation as though it were our own and our assurance processes assign that  
 certiﬁcate precisely the weight it deserves :-)’

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 928 | Ross Anderson |

*28.2. EVALUATION*

*products assurance* (CPA), but the only consumer product for which it currently  
 maintains CPA certiﬁcation is the smart meter discussed in section 14.2.4. Fu-  
 ture legislation will require basic security for IoT devices, including a ban on  
 default passwords and a requirement for a software update mechanism; this is  
 being done in harness with ETSI, leading to a draft European standard ETSI  
 EN 303 645 V2.1 [646].

The direction of travel is now to look at process rather than product, both

for ﬁrms developing critical equipment for Britain’s national infrastructure, and  
 more generally. The general scheme, *Cyber Essentials*, is mandated for govern-  
 ment contractors supplying IT services or handling personal information.

There was already the ISO 27001 standard for security management, which

we mentioned in section 12.2.4: this is expensive, having been turned into an  
 income stream by the big accountancy ﬁrms, and about as useless as CC. Almost  
 all of the large security breaches happen at ﬁrms with ISO 27001 certiﬁcation,  
 where the auditor said something was OK that wasn’t. The auditors have to  
 rely on what the ﬁrms tell them, and a ﬁrm that doesn’t know how to protect  
 its systems will just say ‘We have a great process for X’ when they don’t. Why  
 should a small business owner cough up tens of thousands for that, unless they  
 need it to bid for government contracts? And why should a government impose  
 such a tax? So the Cyber Essentials scheme focuses on the very basic stuff

and costs only £300 for a validated self-certiﬁcation. Its target was small and  
 medium enterprises, but the ﬁrst ﬁrms to be actually certiﬁed under it were  
 large ﬁrms like banks and phone companies who wanted to add every single  
 tassel to their corporate due diligence.

As governments bicker, we’ve seen the emergence of a private sector stan-

dard, Bitsight. Recall how in the ﬁrst chapter I remarked that in the corporate  
 world, a trusted system often means one acceptable to insurers. Recall also how  
 in section 2.2.1.6 we described how the NSA has a system called Mugshot that  
 crawls the Internet looking for vulnerable systems, and another called Xkeyscore  
 that enables cyber-warriors to ﬁnd vulnerable systems near a target of interest?  
 Well, Bitsight does Mugshot for the private sector, but instead of attacking  
 companies’ systems it rates ﬁrms for cybersecurity risk by counting how many  
 of their servers are not patched up to date, and how many other indicators of  
 compromise are visible. They have come to dominate insurance market assess-  
 ments because they give a single numerical rating at a time when the insurance  
 industry, which is cyclical, is having its proﬁts squeezed and can no longer get  
 clients to ﬁll out long questionnaires about their cybersecurity practices. This  
 makes sense in the Lerner-Tirole model, as Bitsight is motivated to keep ahead  
 of possible competitors, just like an elite university. Their ratings are bringing  
 more honesty to the ecosystem than most of the schemes promoted by gov-  
 ernments and audit ﬁrms, but have some interesting side-effects. For example,  
 service ﬁrms are now less willing to sponsor capture-the-ﬂag competitions for  
 schools; if the Bitsight crawler sees a vulnerable system in your IP address space  
 that you set up as a target for such an exercise, it can cut your Bitsight rating  
 by more than 10%, which can cost you real business.

So much for certifying products and business processes. In the next section,

we look more closely at dependability metrics from the viewpoints of failure  
 analysis, bug tracking, cross-product dependencies, open-source software and

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 929 | Ross Anderson |

*28.3. METRICS AND DYNAMICS OF DEPENDABILITY*

the development team.

**28.3** **Metrics and dynamics of dependability**

As dependability becomes a lifetime property we need better ways of measuring  
 it. We know that it is often a function of the development team; we discussed  
 the capability maturity model in section 27.5.3. To get secure code, you need to  
 hire smart people with a suitable mix of skills and get them to work together on  
 shared projects so they learn to work together. In the process, you measure how  
 well they’re doing and improve it by giving feedback and constantly improving  
 the process and tools. But how do you do the measurement?

This has two main aspects: reliability growth, as systems become more

dependable over time with testing and bug ﬁxing, and vulnerability disclosure,  
 as bugs are found and may or may not be ﬁxed.

**28.3.1** **Reliability growth models**

The growth of reliability as systems get more testing, both in the lab and in the  
 ﬁeld, is of interest to many more people than just software engineers; nuclear,  
 electrical and aerospace engineers all depend on reliability models and metrics.

In the simplest possible case – where the tester is trying to ﬁnd a single bug

|  |
| --- |
| in a system – a reasonable model is the Poisson distribution: the probability  *p* that the bug remains undetected after *t* statistically random tests is given  by *p* = *e�Et* where *E* depends on the proportion of possible inputs that it  a↵ects [1175]. So where the reliability of a system is dominated by a single bug  – say when we’re looking for the ﬁrst bug in a system, or the last one – reliability  growth can be exponential. |
| But extensive empirical investigations have shown that in large and complex  systems, the likelihood that the *t*-th test fails is not proportional to *e�Et* but to |
| *k/t* for some constant *k*. So reliability grows very much more slowly. This was  ﬁrst documented in the bug history of IBM mainframe operating systems [18],  and has been conﬁrmed in many other studies [1198]. As a failure probability  of *k/t* means a mean time between failure (MTBF) of about *t/k*, reliability  grows linearly with testing time. This result is often stated by the safety critical  systems community as ‘If you want a mean time between failure of a million  hours, then you have to test for (at least) a million hours’ [355]. This has been  one of the main arguments against the development of complex, critical systems  that can’t be fully tested before use, such as President Reagan’s ‘Star Wars’  ballistic missile defence program. |

The reason for the *k/t* behaviour emerged in [249] and was proved under

more general assumptions by observing that the Maxwell-Boltzmann statistics  
 developed to model ideal gases apply to statistically independent bugs too [312].  
 This model gives a number of other interesting results. If you can assume that  
 the bugs are statistically independent, then the *k/t* reliability growth is the best  
 possible: the rule that you need a million hours of testing to get a million hours  
 MTBF is inescapable, up to some constant multiple which depends on the initial

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 930 | Ross Anderson |

*28.3. METRICS AND DYNAMICS OF DEPENDABILITY*

quality of the code and the scope of the testing. This can be seen as a version  
 of ‘Murphy’s Law’: that the number of defects which survive a selection process  
 is maximised.

These statistics give a neat link between evolutionary models of software and

the evolution of a biological species under selective pressure, where the ‘bugs’  
 are genes that reduce ﬁtness. Just as software testing removes the minimum  
 possible number of bugs consistent with the tests applied, biological evolution  
 enables a species to adapt to a changed environment at a minimum cost in  
 early deaths while preserving as much diversity as possible to help the species  
 survive future environmental shocks. For example, if a population of rabbits  
 is preyed on by snakes, they will be selected for alertness rather than speed.  
 Their variability in speed will remain, so if foxes arrive in the neighbourhood  
 the rabbit population’s average running speed can rise sharply under selective  
 predation9.

The evolutionary model also points to fundamental limits on the reliability

gains to be had from reusable software components such as objects or libraries;  
 well-tested libraries simply mean that overall failure rates will be dominated by  
 new code. It also explains the safety-critical systems community’s observation  
 that test results are often a poor performance indicator [1175]. The failure time  
 measured by a tester depends only on the initial quality of the program, the  
 scope of the testing and the number of tests, so it gives virtually no further  
 information about the program’s likely performance in another environment.  
 There are also some results that are unexpected, but obvious in retrospect: for  
 example, each bug’s contribution to the overall failure rate is independent of  
 whether the code containing it is executed frequently or rarely – intuitively,  
 code that is executed less is also tested less. Finally, different testers should  
 work on a program in parallel rather than in series.

So complex systems only become reliable following prolonged testing by di-

verse testers. This gives the advantage to tried-and-tested designs for machinery,  
 as we gain statistical knowledge of how it fails. Mass-market software started  
 to be used at sufficient scale to enable thorough testing, especially once crash  
 reports started to be sent to the vendor. The use of regression testing by devel-  
 opment teams meant that billions of test cases can be exercised overnight with  
 each new build. Services that move to the cloud can be monitored for failure  
 all the time.

So what are the limits to reliability? First, new bugs are introduced by

the new code in new versions dictated by platform business models, and second,  
 adversarial action brings in a signiﬁcant asymmetry between attack and defence.

Let’s take a simpliﬁed example. Suppose a product such as Windows has

1,000,000 bugs each with an MTBF of 1,000,000,000 hours. Suppose that Ahmed  
 works for the Iranian Revolutionary Guard to create tools to break into the US  
 Army’s network, while Brian is the NSA guy whose job is to stop Ahmed. So  
 he must learn of the bugs before Ahmed does.

Ahmed has only half a dozen people, so he can only do 10,000 hours of testing

a year. Brian has full Windows source code, dozens of PhDs, oversight of the

9More formally, the *fundamental theorem of natural selection* says that a species with a

high genic variance can adapt to a changing environment more quickly [695].

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 931 | Ross Anderson |

*28.3. METRICS AND DYNAMICS OF DEPENDABILITY*

commercial evaluation labs, an inside track on CERT, an information sharing  
 deal with other Five Eyes member states, and also runs the government’s scheme  
 to send round consultants to critical industries such as power and telecomms  
 to ﬁnd out how to hack them (pardon me, to advise them how to protect their  
 systems). This all adds up to the equivalent of 100,000,000 hours a year of

testing.

After a year, Ahmed ﬁnds 10 bugs, while Brian has found 100,000. But the

probability that Brian has found any one of Ahmed’s bugs is only 10%, and the  
 probability that he’ll have found them all is negligible. And Brian’s bug reports  
 will have become such a ﬁrehose that Microsoft will have found some excuse to  
 stop ﬁxing them. In other words, the attacker has thermodynamics on his side.

In real life, vulnerabilities are correlated rather than independent; if 90% of

your vulnerabilities are stack overﬂows, and you introduce compiler technology  
 such as stack canaries and ASLR to trap them, then for modelling purposes there  
 was perhaps only a single vulnerability. However, it’s taken years to sort-of-not-  
 quite ﬁx that one, and new ones come along all the time. So if you are actually  
 responsible for Army security, you can’t just rely on some commercial off-the-  
 shelf product you bought a few years ago. One way to escape the statistical trap  
 is simplicity – which, as we saw in Chapter 9, ends up meaning policies such as  
 mandatory access controls, architecture such as multilevel secure mail guards,  
 and much else besides. The more modern approach is a learning system that  
 observes what’s broken and ﬁxes it quickly. That in turn means vigilant network  
 monitoring, breach reporting, vulnerability disclosure and rapid patching – as  
 we described in section 27.5.7.

**28.3.2** **Hostile review**

When you really want a protection property to hold, it’s vital that the design  
 and implementation be subjected to hostile review. It will be eventually, and  
 it’s likely to be cheaper if it’s done before the system is ﬁelded. As we’ve seen in  
 one case history after another, the motivation of the attacker is critical; friendly  
 reviews, by people who want the system to pass, are essentially useless compared  
 with contributions by people who are seriously trying to break it. That’s the  
 basic reason evaluations paid for by the vendor from one of a number of compet-  
 ing evaluators, as in the Common Criteria and ISO 27001, are fundamentally  
 broken. (Recall our discussion in section 12.2.6 of auditors’ chronic inability to  
 detect fraud by the executives who hired them. One hedge fund manager who  
 made $100M from shorting Wirecard, Jim Chanos, said, “When people ask us,  
 who were the auditors, I always say ‘Who cares?’ Almost every fraud has been  
 audited by a major accounting ﬁrm.” [30].)

To do hostile review, you can motivate attackers with either money or hon-

our. An example of the ﬁrst was the Independent Validation and Veriﬁcation  
 (IV&V) program used by NASA for manned space ﬂight; contractors were hired  
 to trawl through the code and paid a bonus for every bug they found. An ex-  
 ample of the second was in the evaluation of nuclear command and control,  
 where Sandia National Laboratories and the NSA vied to ﬁnd bugs in each  
 others’ designs. Another was at IBM, which maintained a leading position in  
 cryptography for years by having two teams, one in New York and the other in

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 932 | Ross Anderson |

*28.3. METRICS AND DYNAMICS OF DEPENDABILITY*

North Carolina, who would try to break each others’ work, like Cambridge and  
 Oxford trying to win a boat race every year. Yet another is Google’s Project  
 Zero where the company devotes real engineering effort to ﬁnding vulnerabilities  
 both in products that it relies on, such as Linux, and competitor products such  
 as iOS, and aggressively discloses them after 90 days’ notice in order to force  
 them to be ﬁxed. This gets over 97% of them ﬁxed [589].

Review by academics is, at its best, in this category. We academics win

our spurs by breaking stuff, and get the highest accolades by inventing new  
 types of attack. We compete with each other – Cambridge against Berkeley  
 against CMU against the Weizmann. The established best practice, though, is  
 to motivate hostile review with money, and speciﬁcally via bug bounty programs  
 where vendors offer big rewards for reports of vulnerabilities. As we noted in  
 section 27.5.7 above, Apple offers $1m for anyone who can hack the iOS kernel  
 without requiring any clicks by the user; this is one signiﬁcant metric for iOS  
 security10.

One way to turbocharge either academic review or a bug bounty program is

to open your design and implementation, so all the world can look for bugs.

**28.3.3** **Free and open-source software**

Should security mechanisms be open to scrutiny? The historical consensus is  
 that they should be. The ﬁrst book in English on cryptography was written  
 in 1641 by Oliver Cromwell’s cryptographer John Wilkins. In *‘Mercury, or the*  
 *Secret and Swift Messenger’* he justiﬁed discussing cryptography with the re-  
 mark ‘If all those useful Inventions that are liable to abuse, should therefore  
 be concealed, there is not any Art or Science which might be lawfully profest’.  
 The ﬁrst exposition of cryptographic engineering, Auguste Kerckhoffs *‘La Cryp-*  
 *tographie Militaire’* in 1883, recommended that cryptographic systems should  
 be designed in such a way that they are not compromised if the opponent learns  
 the technique being used: security must depend only on the key [1042]. In

Victorian times, the debate also touched on whether locksmiths should discuss  
 vulnerabilities in locks; as I noted in section 13.2.4, one book author pointed out  
 that both locksmiths and burglars knew how to pick locks and it was only the  
 customers who were ignorant. In section 15.8 I discussed the partial openness  
 found even in nuclear security.

The free and open-source software (FOSS) movement extends this philosophy

of openness from the algorithms and architecture to the implementation detail.  
 Many security products have publicly-available source code, of which the ﬁrst  
 was probably the PGP email encryption program. The Linux and FreeBSD

operating systems and the Apache web server are also open-source and are  
 widely relied on: Android runs on Linux, which is also dominant in the world’s  
 data centres, while iOS is based on FreeBSD.

Open-source software is not entirely a recent invention; in the early days

of computing, most system software vendors published their source code. This  
 started to recede in the early 1980s when pressure of litigation led IBM to adopt

10On this metric the most secure system on earth might be bitcoin, as anyone who could

break the signature mechanism could steal billions.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 933 | Ross Anderson |

*28.3. METRICS AND DYNAMICS OF DEPENDABILITY*

an ‘object-code-only’ policy for its mainframe software, despite bitter criticism  
 from its users. The pendulum has swung back since 2000, and IBM is one of  
 the stalwarts of open source.

There are a number of strong arguments in favour of open software, and a

few against. First, while many closed systems are developed in structured ways  
 with waterfall or spiral models of the initial development and later upgrades, the  
 world is moving towards more agile development styles, a tension described by  
 Eric Raymond as “The Cathedral and the Bazaar” in an inﬂuential 1999 book  
 of that name [1584]. Second, systems are getting so complex and toolchains so  
 long that often the bug you’re trying to bust isn’t in the code you wrote but in  
 an operating system or even a compiler on which you rely, so you want to be  
 able to ﬁnd bugs there quickly too, and either get them ﬁxed or contribute a ﬁx  
 yourself. Third, if everyone in the world can inspect and play with the software,  
 then bugs are more likely to be found and ﬁxed; in Raymond’s famous phrase,  
 “To many eyes, all bugs are shallow”. Fourth, it may also be more difficult to  
 insert backdoors into such a product (though people have been caught trying,  
 now that an exploit can sell for seven ﬁgures). Finally, for all these reasons,  
 open source is great for conﬁdence.

The proprietary software industry argues that while openness helps the de-

fenders ﬁnd bugs so they can ﬁx them, it also helps the attackers ﬁnd bugs so  
 they can exploit them. There may not be enough defenders for many open prod-  
 ucts, as the typical volunteer ﬁnds developing code more rewarding than bug  
 hunting (though bug bounties are starting to shift this). Second, as I noted in  
 section 28.3.4, different testers ﬁnd different bugs as their test focus is different.  
 As volunteers will look at cool bits of code such as the crypto, smart spooks or  
 bug-bounty hunters will look at the boring bits such as the device drivers. In  
 practice, major vulnerabilities lurk for years. For example, a programming bug  
 in PGP versions 5 and 6 allowed an attacker to add an extra escrow key without  
 the key holder’s knowledge [1700].

So will the attackers or the defenders be helped more? Under the standard

model of reliability growth, we can show that openness helps attack and defence  
 equally [74]. Thus whether an open or proprietary approach works best in

a given application will depend on whether and how that application departs  
 from the standard assumptions, for example, of independent vulnerabilities. In  
 the end, you have to go out and collect the data; as an example, a study of  
 security bugs found in the OpenBSD operating system revealed that these bugs  
 were signiﬁcantly correlated, which suggests that openness there was a good  
 thing [1488].

So where is the balance of beneﬁt? Eric Raymond’s inﬂuential analysis of

the economics of open source software [1585] suggests ﬁve criteria for whether  
 a product would be likely to beneﬁt from an open source approach: where it  
 is based on common engineering knowledge rather than trade secrets; where  
 it is sensitive to failure; where it needs peer review for veriﬁcation; where it  
 is sufficiently business-critical that different users will cooperate in ﬁnding and  
 removing bugs; and where its economics include strong network effects. Security  
 passes all these tests.

The law-and-economics scholar Peter Swire has explained why governments

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 934 | Ross Anderson |

*28.3. METRICS AND DYNAMICS OF DEPENDABILITY*

are intrinsically less likely to embrace disclosure: although competitive forces  
 drove even Microsoft to open up a lot of its software for interoperability and  
 trust reasons, government agencies play different games, such as expanding their  
 budgets and avoiding embarrassment [1853]. Yet even there, the security argu-  
 ments have started to prevail: from tentative beginnings in about 1999, the US  
 Department of Defense has started to embrace open source, notably through  
 the SELinux project I discussed in section 9.5.2.

So while an open design is neither necessary nor sufficient, it is often going

to be helpful. The important ﬁrst-order questions are how much effort was ex-  
 pended by capable people in checking and testing what you built – and whether  
 they tell you everything they ﬁnd. The prudent thing to do here is to have a  
 generous bug-bounty program. And there’s a second-order question of grow-  
 ing importance: if your business depends on Linux, shouldn’t at least a couple  
 of your engineers be engaged its its developer community, so you know what’s  
 going on?

**28.3.4** **Process assurance**

In recent years less emphasis has come to be placed on assurance measures  
 focused on the product, such as testing, and more on process measures such as  
 who developed it and how. As anyone who’s done system development knows,  
 some programmers produce code with an order of magnitude fewer bugs than  
 others. There are also some organizations that produce much better code than  
 others. Capable ﬁrms try to hire good people, while good people prefer to work  
 for ﬁrms that value them and that hire kindred spirits.

While some of the differences between high-quality and low-quality devel-

opers are down to talent, many are conditioned by work culture. In my own  
 experience, some IT departments are slow and bureaucratic while others are  
 lively. Leadership matters; just as replacing Boeing’s engineering leadership

with money men contributed to the 737Max disaster, I’ve seen an IT depart-  
 ment’s morale collapse when its CIO was replaced by a bureaucrat. Another  
 problem is that engineer quality has a tendency to decline over time. One fac-  
 tor is glamour: a lot of bright graduates want to work for startups rather than  
 the big tech ﬁrms, or for racy ﬁntechs and hedge funds rather than boring old  
 money-centre banks. Another is demographics: the Microsoft of the early 1990s  
 was full of young engineers working long hours, but a decade later many had  
 cashed their stock options and left, while the rest had mostly acquired families  
 and worked office hours. Once a company stops growing, promotion is slow;  
 there was a saying in IBM that ‘The only people who ever left were the good  
 ones11.’ Banks and government agencies have similar problems. Some ﬁrms  
 have tried to counter this by rating systems that require managers to ﬁre the  
 least productive 10% or so of their team each year, but the damage this does  
 to morale is dreadful; people spend their time sucking up rather than writing  
 code. Maintaining a productive work culture is one of the really hard problems  
 and a surprising number of big-name ﬁrms are really bad at it. The capability  
 maturity model, which we discussed in section 27.5.3, is one of the tools that can  
 help good managers keep good teams together and improve them over time. But

11As a former IBM employee, I liked that one!

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 935 | Ross Anderson |

*28.3. METRICS AND DYNAMICS OF DEPENDABILITY*

on its own it’s not enough. The whole corporate environment matters, from the  
 water-cooler chat to the top leadership. Is the mission to do great engineering,  
 or just to make money for Wall Street? Of course every ﬁrm pretends to have  
 a mission, but most are bogus and the staff see through them instantly.

Some old-fashioned companies swear by the ISO 9001 standard, which re-

quires them to document their processes for design, development, testing, docu-  
 mentation, audit and management control generally. For more detail, see [1937];  
 a whole industry of consultants and auditors has got its snouts in this trough.  
 Like ISO 27001 which we discussed in section 28.2.9 above, it’s decorative rather  
 than effective. At best it can provide a framework for incremental process im-  
 provement; but very often it’s an exercise in box-ticking that merely replaces  
 chaos by more bureaucratic chaos. Just as agile development methodologies

displaced waterfall approaches, so ISO 9001 is being displaced by the capabil-  
 ity maturity model. What that comes down to, in assurance terms, is trusted  
 suppliers.

But trusted suppliers are hard to certify. Government certiﬁers cannot be

seen to discriminate, so a program degenerates into box-ticking. Private certiﬁ-  
 cation schemes have a tendency to reinforce cartels, or to race to the bottom, as  
 we discussed above in section 28.2.8. In both cases the consultancies and audit  
 ﬁrms industrialise the process to maximise their fee income, and we get back  
 to where we started. If you are good at your job, how do you get that across?  
 Small businesses who do high-quality work generally do better when they sell to  
 the most discriminating customers – to the few big players who’re smart enough  
 to appreciate what they do. In short, you usually have to be an expert yourself  
 to really understand who the quality providers are.

So what about the dynamics? If quality is hard to measure, and the in-

centives for quality are mixed, and improving quality is hard, then what can  
 usefully be said about the assurance level of evolving products? Will they be  
 like milk, or like wine [1488]? Will they get better with age, or go off?

The simple answer is that you have to do real measurements. The quality

of a system may improve, or decline. It may even ﬁnd an equilibrium if the  
 rate at which new bugs are introduced by product enhancements equals the  
 rate at which old bugs are found and removed. There are several research

communities measuring reliability, availability and maintainability of systems  
 in various applications and contexts. Empirically, the reliability of new systems  
 often improves for a while as the more energetic bugs are found and ﬁxed, then  
 stays in equilibrium for a number of years, and then deteriorates as the code  
 gets complex and more difficult to maintain (which software engineers sometimes  
 even refer to as *senescence*). However, if the ﬁrms that maintain the code are  
 still making enough money from it, and are incentivised to care about quality,  
 they can ﬁx this by rewriting the parts that have become too messy – a process  
 known as *refactoring*. In short, the real world is complicated. Models can take  
 you only so far, and you have to study how a system behaves in actual use.

Measurement brings its own problems. Some vendors collect and analyse

masses of data about how their products fail – examples being platform compa-  
 nies like Microsoft, Google and Apple – but make only selected data available  
 to outsiders, creating a market for specialist third-party evaluators, from the

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 936 | Ross Anderson |

*28.4. THE ENTANGLEMENT OF SAFETY AND SECURITY*

tech press to academics. Other ﬁrms say much less, creating an opportunity for  
 rating ﬁrms such as Bitsight. The healthcare sector is notoriously cagey about  
 evidence of harm to patients, whose lawyers may have to work for years to build  
 a negligence case. But in applications such as medical devices, there is enough  
 of a public interest for regulators to intervene to increase transparency, and as  
 we noted in section 28.2.3 above, the EU recently changed the law on medical  
 device regulation to compel aftermarket surveillance. As most software nowa-  
 days is in applications rather than platforms, and very often in or supporting  
 devices, this brings us to consider the regulation of safety.

**28.4** **The Entanglement of Safety and Security**

As we discussed in 28.2.2 governments regulate safety for many types of device  
 from cars to railway signals and from medical devices to toys. As software ﬁnds  
 its way into everything and everything gets connected to cloud services, the  
 nature of safety regulation is changing, from simple pre-market safety testing  
 to maintaining security and safety over a service lifetime of years during which  
 software will be patched regularly. We’ve already seen how this is becoming  
 entangled with security. We discussed smart grids in section 23.8.1, smart meters  
 in section 14.2 and building alarms in section 13.3.

I believe that the increasing entanglement of safety and security is so signif-

icant for our ﬁeld that since 2017 we’ve merged teaching on safety and security  
 for our ﬁrst-year undergraduates, as I mentioned in section 27.1. Safety is a  
 much more diverse subject than security. While security engineering is a fairly  
 coherent discipline, safety engineering has fragmented over time into separate  
 disciplines for aircraft, road vehicles, ships, medical devices, railway signals and  
 other applications. We can still learn a lot from safety engineers, as I discussed  
 in section 27.3, and safety engineers are starting to have to learn about security  
 too. This will be a long process. Thanks to the coronavirus lockdown, these  
 lectures are now publicly available on video [89]; I now wish I’d put my lectures  
 online years ago.

What spurred us to unite security and safety teaching was some work we

did for the European Union in 2015–6 looking at what will happen to safety  
 regulation once computers are embedded invisibly everywhere. The EU is the  
 leading safety regulator worldwide for dozens of industries, as it’s the largest  
 market and cares more about safety than the US government does. Officials  
 wanted to know how this ecosystem would have to adapt to the ‘Internet of  
 Things’ where vulnerabilities (whether old or new) may be remotely exploited,  
 and at scale. Many regulators who previously thought only in terms of safety  
 will have to start thinking of security as well.

The problem facing the EU in 2015 was how to modernise safety regulation

across dozens of industries from cars and planes to medical devices, railway  
 signals and toys, and to introduce security regulation as appropriate. The reg-  
 ulatory goals are different. In this book, we have discussed how security fails in  
 a number of different sectors and the nature of the underlying market failure.  
 In different contexts, security regulators might want to drive up attackers’ costs  
 and reduce their income; to reduce the cost of defence; to reduce the impact of

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 937 | Ross Anderson |

*28.4. THE ENTANGLEMENT OF SAFETY AND SECURITY*

security failure; to enable insurers to price cyber-risks efficiently; and to reduce  
 both the social cost of attacks and social vulnerability to them.

Safety regulators seem to be more straightforward. They tend to ignore

the economic subtleties underlying each market failure and focus on injury and  
 death, then on direct property damage. For deaths, at least, you’d think we  
 have decent statistics, but priorities are modulated by public concern about  
 different types of harm. As we’ve discussed, the public are much more alarmed  
 at a hundred people dying all at once in a plane crash than a thousand people  
 dying one at a time in medical device accidents. However, when hackers showed  
 they could go in over wiﬁ and change the dose delivered by several models  
 of Hospira Symbiq infusion pump to a potentially fatal level, the FDA issued a  
 safety advisory telling hospitals to stop using it [2066]. It did not issue advisories  
 about the 300+ models that merely suffered from the safety issues we discussed  
 in section 28.2.3. When you stop to think about it, that’s rather striking. A  
 safety regulator ignores a problem that kills several thousand Americans a year  
 while panicking at a safety-plus-security issue that has so far killed nobody.  
 Perhaps people intuitively grasp the principle we discussed in section 27.3.6:  
 that a one-in-a-million chance of a fatal accident happening by chance doesn’t  
 give much assurance if an opponent can engineer the combination of inputs  
 needed to trigger it.

The pattern continued the following year, when the FDA recalled 465,000

St Jude pacemakers in the USA for a ﬁrmware update after a report that the  
 device could be hacked. The update involves a hospital visit because of a small  
 risk of device failure. The report itself was controversial, as it was promoted by  
 an investment ﬁrm that had shorted St Jude’s stock [1838].

The EU already had work in progress on medical device safety and, the

following year, updated its Medical Device Directives to require that medical  
 device software be developed ‘in accordance with the state of the art taking into  
 account the principles of development life cycle, risk management, including in-  
 formation security, veriﬁcation and validation’, and ‘designed and manufactured  
 in such a way as to protect, as far as possible, against unauthorised access that  
 could hamper the device from functioning as intended’ [653]. This text doesn’t  
 cover all the bases but is a useful ﬁrst step; it comes into force in 2021.

**28.4.1** **The electronic safety and security of cars**

Road safety helped drive interest in the convergence of security and safety in  
 the mid-2010s, thanks to the surge of interest in self-driving cars driven by  
 Google and Tesla, among others. Following the breakthrough in computer vision  
 using deep neural networks in 2012, there was rapid progress. The ﬁrst news  
 of early accidents with experimental vehicles arrived around 2015 at the same  
 time as the breakthrough research on adversarial machine learning I described  
 in section 25.3 and the high-proﬁle hack of the Jeep Cherokee, which I described  
 in section 25.2.4. Autonomous cars suddenly became a hot topic, not just for  
 stock-market investors and security researchers, but for safety. Could terrorists  
 hack them and drive them into crowds? Could they get the same result by

projecting deceptive images on a building? And if kids could use their phone  
 to hail a car home from school, could someone hack it to abduct them? And

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 938 | Ross Anderson |

*28.4. THE ENTANGLEMENT OF SAFETY AND SECURITY*

what about the ethics – if a self-driving car was about to crash and could choose  
 between killing its one occupant or two pedestrians, what would it do? What  
 should it do? Let’s take the safety and assurance aspects one step at a time.

Road safety is a major success story for safety regulation. Following Ralph

Nader’s book ‘Unsafe at any speed’ [1370], the US Congress created the National  
 Highway Traffic Safety Administration (NHTSA). It started from a belief that  
 crash testing of new models would be enough, but found it needed to force the  
 recall of vehicles that were discovered later to be unsafe12. The effects can be  
 seen starkly in a Consumer Reports video of a crash test between a 2009 Chevy  
 Malibu and a 1959 Chevy Bel Air. The Bel Air’s passenger compartment is  
 crushed and the dummy driver impaled on the steering wheel; a human driver  
 would have been killed. Thanks to 50 years of progress, the passenger compart-  
 ment of the Malibu remains intact; the front crumple zone absorbs much of the  
 energy, the seatbelt and airbag hold the dummy driver, and a human driver  
 would have walked away [472]. I show this video to my ﬁrst-year students to  
 emphasise that safety engineering is not just about making mistakes less likely,  
 but also about mitigating their effects. The decades of progress that the video  
 illustrates involved not just engineering, lobbying and standard setting across  
 multiple countries, but many tussles between safety campaigners and the indus-  
 try. Within the industry, some carmakers tried to lead while others dragged their  
 heels. Car safety also involves driver training, laws against drink driving and  
 excessive driver working hours, changing social norms around such behaviour,  
 steady improvements to road junction design and much else. It has grown into  
 a large and complex ecosystem. This now has to evolve as cars become smarter  
 and more connected.

During the 2010s, cars were steadily acquiring more assistive technology,

from parking assist through adaptive cruise control to automatic emergency  
 braking and automatic lane keeping. I described in section 25.2 how compa-  
 nies like Google and Tesla drove a research program to join these systems up  
 together, giving autonomous driving. The assistive technology features them-  
 selves had various bugs; I discussed the blind spots of adaptive cruise control in  
 section 23.4.1. Some were also open to exploitation: Charlie Miller and Chris  
 Valasek had hacked the Jeep’s park-assist feature to drive it off the road. Com-  
 panies that sold limited autonomous driving features, such as Tesla, experienced  
 accidents that began to undermine public conﬁdence. I discussed some of the  
 security implications of autonomous vehicles in section 25.2. We discussed the  
 usability aspects of safety too. Tesla’s ‘Autopilot’ required the driver to pay  
 attention and keep a hand on the steering wheel, in order to remain in control  
 and avoid accidents. But as it drove adequately much of the time, many drivers  
 didn’t, with consequences that were occasionally both fatal and newsworthy.  
 Even in 2020, while the better autopilot systems can drive a car passably well  
 on the motorway, they can be ﬂaky on smaller roads, getting confused at round-  
 abouts and running over grass verges. So how should we test their safety?

Testing an *anti-lock braking system* (ABS) is fairly straightforward as we

understand the physics of skidding and aquaplaning, and such systems have  
 been around long enough for us to have a long accident history. We next had  
 *emergency brake assist* (EBA), which applies full braking force if it thinks you’re

12The story is told in ‘The Struggle for Auto Safety’ [1235].

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 939 | Ross Anderson |

*28.4. THE ENTANGLEMENT OF SAFETY AND SECURITY*

trying to do an emergency stop. The usual algorithm is that if you move your  
 foot from the accelerator to the brake in under 300ms and then apply at least  
 2kg of force, it activates and stops the car as quickly as possible. This is a  
 simple algorithm but is harder to evaluate, as it’s trying to infer the driver’s  
 intent. (I once triggered mine unintentionally and thankfully there wasn’t a car  
 close behind me.)

A recent addition is automatic emergency braking (AEB) which is supposed

to stop the car if a child or a dog runs in front of you. This is harder still,  
 as you’re trying to understand everything you see on the street ahead, with  
 complex processing that uses both traditional logic and machine-vision systems  
 based on deep neural networks. As we discussed in section 25.2, the current  
 products are both limited and of variable quality. Add lane keeping assist and  
 adaptive cruise control, and your car can pretty well drive itself on the freeway.  
 But how should you test that? And if we ever move to full autonomy, your risk  
 and threat analysis must include a lot of the bad things that happen in human  
 societies.

Tesla says in defence of its Autopilot feature that its cars are safer than

others; of the 135 fatalities in crashes involving its vehicles up to June 23 2020,  
 only 10 were attributed to Autopilot [1870]. The actual ﬁgures are controversial,  
 though. An insurance forensics company brought a lawsuit against NHTSA to  
 get the raw ﬁgures for accidents up till June 2016, studied them, and claimed  
 that the analysis offered by Tesla and accepted by NHTSA had considered only  
 13% of the data. Rather than a 40% decrease in airbag deployments after the  
 Autosteer feature of the vehicle was activated, as Tesla had claimed, the full  
 data showed a 57% increase from 0.76 deployments per million miles of travel  
 to 1.21 [1565].

The insurance industry accumulates good data over time across all car mak-

ers and worries about the cost of claims. It was concerned at AEB, worrying  
 that if cars brake hard when a rabbit runs in front of them, there might be  
 more rear-end collisions. But once the data started to arrive in 2016, insurers  
 relaxed. When I check online how much it would cost me to insure a Tesla with  
 Autopilot versus a plug-in hybrid Mercedes of similar value, I get about the  
 same answer (though more insurers bid for the Mercedes).

But actuarial costs are not the only driver of public policy. Politicians started

to worry about truck drivers’ jobs. Philosophers started to worry about ethics:  
 given a choice between killing a pedestrian and the driver, would an autopilot  
 protect its driver? The industry worried about updates. Progress in machine  
 vision is so rapid that you can imagine having to sell a whole new vision unit  
 every ﬁve years, as the systems we have now won’t run on the hardware of ﬁve  
 years ago. Would the customers stand for having to pay several thousand Euros  
 every few years for a new autopilot?

People also worry more about security threats, as we have evolved to be

sensitive to adversarial activity. By 2020, we have a ﬂurry of security stan-

|  |
| --- |
| dardisation, including the draft ISO 21434 standard on cybersecurity, which I  mentioned in section 27.3.5; proposed amendments to the regulations of the UNECE13 to deal with cybersecurity and software updates for connected vehi- |

13The UN Economic Commission for Europe was established by a 1958 treaty. It includes

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 940 | Ross Anderson |

*28.4. THE ENTANGLEMENT OF SAFETY AND SECURITY*

cles [1921]; and in Japan, following cyber attacks on Toyota and Honda, baseline  
 requirements for the whole car industry supply chain [1243]. That’s all great,  
 but the target is moving faster all the time.

In Brussels, officials started to worry about how the regulatory ecosystem

could cope. Over 20 agencies are involved one way or another in vehicle safety  
 (unlike in the USA, where NHTSA covers everything from car design to speed  
 limits). Would each agency have to hire a security engineer? Some of them don’t  
 have any engineers at all, just lawyers and economists. How should the ecosys-  
 tem evolve to cope? Officials were suddenly less willing to trust the industry’s  
 assurances after the Dieselgate emissions scandal in 2015, when it turned out  
 that Volkswagen had installed software in its cars to cheat on emissions tests.  
 The Volkswagen and Audi CEOs lost their jobs and face criminal charges, along  
 with about a dozen other executives; the companies paid billions in legal settle-  
 ments. The threat model was no longer just the external hacker, but included  
 the vendors themselves. Regulators wanted to get back in control. What did  
 they need to do?

**28.4.2** **Modernising safety and security regulation**

|  |
| --- |
| Our brief was to consider the policy problem generally across all sectors. It  was clear that European institutions needed cybersecurity expertise to support |
| safety, privacy, consumer protection and competition. But what would this  mean in practice? In order to ﬂesh this out, ´Eireann Leverett, Richard Clayton |
| and I studied three industries of which we had some knowledge: medical devices,  cars and electricity distribution. Our full report [157] was presented in 2016 and  published the following year, along with a summary version for academic audi-  ences [1148]. The full report has an extensive analysis of the existing patchwork  of safety / security standards for embedded devices from ISO, IEC, NIST and  others. |

This exercise taught us a huge amount about subjects we didn’t expect would

be on the agenda. Usability is critical in a number of ways. The dominant safety  
 paradigm used to be to analyse how limited or erratic human performance could  
 degrade an otherwise well-designed system, and then work out how to mitigate  
 the consequences. Some countries demand that drivers over 67 get a medical  
 or re-sit their driving test, as well as insisting on seat belts and airbags. In  
 security, malice comes into the equation: you worry about the widow in her  
 eighties who’s called up and persuaded to install an ‘upgrade’ on her PC. Car  
 security is not just about whether a terrorist can take over your car remotely  
 and drive it into some pedestrians. If a child can use her mobile phone to direct  
 a car to take her to school, what new threats do we have to worry about? Might  
 she be abducted, whether by a stranger or (more likely) in a custody dispute?  
 And whose engineers need to worry about her safety – the car company’s, the  
 ride-hailing company’s, or the government’s?

The security engineer’s task is to enable even vulnerable users to enjoy rea-

sonable protection against a capable motivated opponent. How do you embed

the car-making countries in Europe and Africa plus Japan, Korea and Australasia and is  
 effectively one of three standardisation zones for cars, the others being the Americas and  
 China.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 941 | Ross Anderson |

*28.4. THE ENTANGLEMENT OF SAFETY AND SECURITY*

good practice in industries that have never had to think of distant adversaries  
 before? That’s not just a matter of setting minimum standards but also of

embedding security thinking into standards bodies, regulatory agencies, testing  
 facilities and many other places in the ecosystem. That will be a long and ardu-  
 ous process, just as car safety was. Getting test engineers who work by checking  
 carefully whether the ‘British standard ﬁnger’ can be accidentally poked into  
 an electrical appliance to think in terms of creative malice instead will be hard.  
 Where do we start?

We came up with a number of recommendations. Some were considered by

the Commission to be in the ‘too hard’ category, including extending product  
 liability law to services, and requiring the reporting of breaches and vulnerabili-  
 ties not just to security agencies and privacy regulators but to other stakeholders  
 too. Eventually we’ll need laws regulating the use of car data in investigating  
 accidents, particularly if there are disputes over liability when car autopilots  
 cause fatal crashes. (At present the vendors hold the data close and it takes  
 vigorous litigation to get hold of it.) Without data we won’t be able to build a  
 learning system.

One of our recommendations was that vendors should have to self-certify, for

their CE mark, that products can be patched if need be. This looks set to be  
 partly achieved by means of a technical standard, ETSI EN 303 645 V2.1 [646],  
 as I discussed in section 28.2.9 above. ETSI is a membership organisation of  
 some 800 ﬁrms; it can move more quickly than governments but still has some  
 clout; for example, it set up the standards bodies for mobile telephony. Failure  
 to comply with an ETSI standard does not however empower a customs officer  
 in Rotterdam to send a container of toys back to China. For that, we need to  
 endow standards with the force of law.

**28.4.3** **The Cybersecurity Act 2019**

Another recommendation was that Europe should create a European Security  
 Engineering Agency to support policymakers. Europe already had the European  
 Network and Information Security Agency (ENISA) which coordinated security  
 breach reporting among EU government agencies, but it had been exiled to  
 Crete as a result of lobbying by the UK and French intelligence agencies, who  
 did not want a peer competitor among the European institutions. The Brexit  
 vote shifted the politics and made it feasible for ENISA to open a proper Brussels  
 office so it could take on the security engineering advisory role.

The Cybersecurity Act 2019 formalised this [655]. It empowered ENISA

to be the central agency for regulating security standards, as we described in  
 section 28.2.9, and also to be the main agency for cybersecurity advice to other  
 European bodies. It is to be hoped that ENISA will build its competence and  
 clout over time, and see to it that new safety standards pay appropriate attention  
 to security too, including at a minimum an appropriate development lifecycle  
 (which was another of our recommendations).

For a security technology to really work, functionality isn’t enough, and the

same goes for testing and even incentives for learning. The right people have to  
 trust it and it has to become embedded in social and organisational processes,

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 942 | Ross Anderson |

*28.5. SUSTAINABILITY*

which means alignment with wider systems and stable persistence over a long  
 enough period of time. The implication is that regulators should shift from

the testing of products to the assurance of whole systems (this was our ﬁnal  
 recommendation).

**28.5** **Sustainability**

The problem our report identiﬁed as the most serious in the long term was that  
 products are becoming much less static. As security and safety vulnerabilities  
 are patched, regulators will have to deal with a moving target. Automobile

mechanisms will need security testing as well as safety testing, and also means  
 of dealing with updates. As we saw from the Volkswagen debacle, many legacy  
 manufacturers haven’t caught up with coordinated disclosure.

Most two-year old phones don’t get patched because the OEM and the mobile

network operator can’t get their act together. So how on earth are we going to  
 patch a 25-year-old Land Rover that spent 10 years in the Danish countryside  
 and was then exported to Romania? This kicked off a political ﬁght, as the car  
 industry did not want to be liable for software patching for more than six years.  
 (The typical European car dealer will sell you a 3-year lease on a new car if  
 you’re rich, and on an approved used car if you’re not quite so rich.) However,  
 the embedded carbon cost of a new car – the amount of CO2 emitted during  
 its manufacture – is about equal to its lifetime fuel burn. And it’s predictable  
 that, sooner or later, a car whose software isn’t up-to-date won’t be allowed on  
 the roads. At present, the average age of a car at scrappage is about 15 years;  
 if that were reduced to six, the environmental cost would be unacceptable. We  
 would not even save CO2 by moving from internal combustion engines to electric  
 vehicles, because of the higher embedded carbon cost of electric vehicles; the  
 whole energy transition is based on the assumption that they will last at least  
 as long as the 150,000km average of our legacy ﬂeet [614].

We found a very ready audience in European institutions. A number of

other stakeholders had been complaining about the effects of software on the  
 durability of consumer goods, with updates available only for a short period  
 of time or not at all. Right-to-repair activists were campaigning for consumer  
 electronic devices to be reusable in a circular economy, annoyed that tech ﬁrms  
 try to prevent repair using ‘security’ mechanisms, or even abuse them in an  
 attempt to make repair illegal. The self-regulation of the IoT market has been  
 largely unsuccessful, thanks to a complex interplay of economic incentives and  
 consumer expectations [1954]. Consumer-rights organisations were starting to  
 warn of the shockingly short lifespan of smart devices: you could spend extra on  
 a ‘smart fridge’ only to ﬁnd that it turned into a frosty brick a year later when  
 the vendor stopped maintaining the server [933]. Planned obsolescence was

already a hot political topic as green parties increased their vote share across  
 Europe. Lightbulbs used to last longer; the bicentennial light has been burning  
 at Livermore since 1901. In 1924 a cartel of GE, Osram and Philips agreed  
 to reduce average bulb lifetimes from 2500h to 1000h, and this behaviour has  
 been followed by many industries since. Governments have pushed back; France  
 made it illegal to shorten product life in 2015, and after Apple admitted in 2017

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 943 | Ross Anderson |

*28.5. SUSTAINABILITY*

that it had used a software update to slow down older iPhones, prompting users  
 to buy newer ones, it was prosecuted. In 2020 it received the highest-ever ﬁne,  
 e1.2B, for anti-competitive practices, although this also related to its treatment  
 of its French distributors [1193]. (It settled a US class action for $500m [966].)

Security agencies were already warning us about the risks of the ‘Internet

of Things’, including network-connected devices with default passwords and  
 unpatchable software. In fact, I learned of the Mirai botnet taking down Twitter  
 as I was on the Eurostar train back to London from giving the ﬁrst presentation  
 of our work, to an audience of about 100 security and IT policy people in  
 Brussels. We soon found out that it exploited Xiaomi CCTV cameras that had  
 default passwords and whose software could not be patched. It was a perfect  
 illustration of the need for action.

Over the ensuing three years there was more than one initiative to try to

create a legal means to push back on tech companies that failed like Xiaomi  
 to support their products by patching vulnerabilities (or even making patching  
 possible). The tech lobbyists blocked the ﬁrst couple of attempts, but eventu-  
 ally in 2019 the European Parliament updated consumer law to cover software  
 maintenance.

**28.5.1** **The Sales of Goods Directive**

This Directive passed the European Parliament in May 2019 [656] and will take  
 effect from 2021. Thereafter, ﬁrms selling goods ‘with digital elements’ must  
 maintain those elements for a reasonable service life. The wording is designed  
 to cover software in the goods themselves, online services to which the goods  
 are connected, and apps which may communicate with the goods either via the  
 services or directly. They must be maintained for a minimum of two years after  
 sale, and for a longer period if that is a reasonable expectation of the customer.  
 What might that mean in practice?

Existing regulations require vendors of durables such as cars and washing

machines to keep supplying spares for at least ten years, so we can hope that  
 the new regulatory regime will require at least as long. Indeed, the preamble to  
 the Directive notes that “A consumer would normally expect to receive updates  
 for at least as long as the period during which the seller is liable for a lack of  
 conformity, while in some cases the consumer’s reasonable expectation could  
 extend beyond that period, as might be the case particularly with regard to  
 security updates.” Given that in many countries cars have to pass an annual  
 roadworthiness test to remain in use, and that such a test is likely to include  
 a check that software is patched up to date in the foreseeable future, we could  
 well see a requirement for security patches to extend beyond ten years.

No doubt there will be all sorts of arguments as the lobbyists try to cut the

costs of this, but it’s a huge step in the right direction. American practice often  
 follows Europe on safety matters.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 944 | Ross Anderson |

*28.5. SUSTAINABILITY*

**28.5.2** **New research directions**

Now that there is not just a clear social need for long-term maintenance of the  
 safety and security of software in durable goods, but a clear legal mandate, I urge  
 my fellow computer scientists to adopt this as a grand challenge for research.

Since the 1960s we have come to see computers almost as consumables,

thanks to Moore’s law. This has conditioned our thinking from the lowest level  
 of technical detail up to the highest levels of policy. We’ve crammed thou-

sands, and then millions, more transistors into chips to support more elaborate  
 pipelining and caching. We’ve put up with slow and inefficient software in the  
 knowledge that next year’s PC will run it faster. We’ve shrugged off monop-  
 olies, believing that the tech ten years from now will be quite different from  
 today’s, so we can replace competition in the market with competition for the  
 market. We’ve been like a cruise ship, happily throwing the trash overboard in  
 the expectation that we’ll leave it far behind us.

Moore’s law is now running out of steam. The analysis of CPU performance

by Hennessy and Paterson shows that while this grew by 25% per annum from  
 1978 to 1986 and a whopping 52% from 1986 to 2003, it slowed to 23% in  
 2003–11, 12% in 2013–15 and 3.5% after that [882]. As the party winds down,  
 we’ll have to start clearing up the trash. That extends from the side-channel  
 attacks like Spectre that were caused by the 12-stage CPU pipelines, through  
 the technical debt accumulated in our bloatware, right up to the monopolistic  
 business ecosystem that drives it all.

There is much, much more. The root certiﬁcates of a number of popular

CAs are starting to expire, and if these are embedded in devices such as TVs  
 whose software can’t be upgraded, then the devices are essentially bricked [117].  
 (The most popular, Letsencrypt, rolls over in 2021.) When CA root certs expire  
 you have to update clients, not servers, to ﬁx them. In consumer devices, the  
 trend is towards shorter lifetimes, to make crypto updateable; as I discussed in  
 section 21.6, browsers such as Safari and Chrome are starting to enforce 398-day  
 certiﬁcate expiry, and that’s another strong incentive for frequent updates.

There are many environments with long-lived equipment where updates

aren’t usual, from petrochemical plants to electricity substations. Systems in  
 buildings and civil engineering projects are somewhat of a hybrid; some vendors  
 are working on versions of Linux that are expected to be as stable as possible  
 and maintained for 25 years, while others are pushing for more aggressive reg-  
 ular updating of whole systems and telling us to ‘put everything in the cloud’.  
 This latter approach is associated with the ‘smart buildings’ meme, but has  
 its own drawbacks. Once multiple contractors and subcontractors need online  
 access to systems that contain full engineering information on buildings – from  
 the electricity substations through the air-conditioning to the ﬁre and burglar  
 alarms – there are obvious risks. Some of these contractors operate at interna-  
 tional scale, so a subverted employee or rooted machine there may have access  
 to the critical national infrastructure of dozens of countries. Are we comfortable  
 with that?

Adapting to the new normal will take years, as it will require behaviour

change by millions of stakeholders. I suspect that the tensions created by this

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 945 | Ross Anderson |

*28.5. SUSTAINABILITY*

adaption will become signiﬁcant in policy, entrepreneurship and research over  
 the next decade.

So what might sustainable security research look like? As a ﬁrst pilot project,

Laurent Simon, David Chisnall and I tackled the maintenance of cryptography  
 software. As I mentioned in section 19.4.1, TLS was proven secure twenty

years ago but there’s been about one attack a year on it since, mostly via  
 side channels. One of the problems is that the crypto implementation, such as  
 OpenSSL, typically has code designed to perform cryptographic operations in  
 constant time, so that the key in use won’t leak to an outside observer, and also  
 to zeroise memory locations containing key material or other sensitive data,  
 so that the key can’t be deduced by other users of the same machine either.  
 But every so often, somebody improves a compiler so that it now understands  
 that certain instructions don’t do any real work. It optimises them away, and  
 all of a sudden millions of machines have insecure crypto software. This is

extremely annoying; you’re out there ﬁghting the bad guys and all of a sudden  
 your compiler writer stabs you in the back, like a subversive ﬁfth column in your  
 rear. Our toolsmiths should be our allies rather than our enemies, and so we  
 worked out what would be needed to ﬁx this properly. Languages like C have no  
 way of expressing programmer intent, so we ﬁgured out how to do this by means  
 of code annotations. Getting a compiler to do constant-time code and secure  
 object deletion properly turns out to be surprisingly tricky, but we eventually  
 got a working proof of concept in the form of plugins for LLVM [1758].

Much, much more will be needed. Moving from the low level of compiler

internals to the medium level of safety systems, a big challenge facing the car  
 industry is getting accident data to the stakeholders who can learn from it. In  
 Europe, some ﬁfty thousand people die in road traffic accidents each year, and  
 another half a million are injured. Worldwide, there are something like a million  
 deaths a year. As cars are starting to log both control inputs and sensor data,  
 there are many megabytes of data about a typical accident, but at present these  
 are mostly not analysed. Increasingly, the data are on the vendors’ servers as  
 well as in the damaged vehicles. But when the police investigate major road  
 accidents, they do not at present have access to much information from data  
 recorders or to most of the 100-million-plus lines of software in the vehicle –  
 some of which will be from subsidiary suppliers, and of uncertain provenance,  
 version and patch status. Where there is a closely-fought lawsuit, data may be  
 demanded, but vendors are reluctant to share it and it typically takes a court  
 order.

What should happen? We should aim at a learning system. We keep hear-

ing reports of people getting killed by an autonomous car in a stupid accident  
 – as when an Uber killed Elaine Herzberg in Tempe, Arizona because she was  
 pushing a bike on the road and its software detected pedestrians only on or  
 near a crosswalk [1264]. We should expect to be able to push an update to stop  
 that happening again. So what would the patch cycle look like? In aviation,  
 accidents are monitored resulting in feedback not just to operators such as pi-  
 lots and air traffic controllers but to the designers of aircraft and supporting  
 ground systems. Work is starting on systems for monitoring accidents involving  
 medical devices, though the vendors may well drag their feet. There, too, the  
 key is mandatory systems for monitoring adverse events and collecting data.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 946 | Ross Anderson |

*28.6. SUMMARY*

At present, we ﬁx road junctions once there have been several accidents there;  
 that’s all the ‘patch cycle’ we have at present, because the only data available  
 to the highways department is the location and severity of each accident, plus  
 perhaps a couple of sentences in the report from the attending officer. A learn-  
 ing system for cars too is inevitable as vehicles become more autonomous, but  
 they won’t learn on their own.

Learning will involve analysing the causes of failures, accumulating engineer-

ing knowledge, and ultimately politics involving multiple stakeholder groups.  
 For starters, we’ll need the ﬁne-grained data from what the cars sensed, what  
 they decided to do, and why. The task of writing the laws to get these data  
 from vendors to accident investigators, insurance assessors and other stakehold-  
 ers lies ahead. At present, EU Member States are responsible for post-market  
 surveillance of vehicle standards, so very little gets done, and there have been  
 proposals to give the European Commission a surveillance power in the wake of  
 Dieselgate. Then there will be the task of actually building these systems. They  
 will be large and complex, because of the need to deal with multiple conﬂicting  
 rights around safety, privacy and jurisdiction.

Moving still further up the stack to the level of policy, there’s a growing

consensus that tech needs to be better regulated. We could perhaps tolerate the  
 various harms to privacy and competition while the technology was changing  
 rapidly. If you didn’t like the IBM monopoly in the 1980s you just had to wait  
 until Microsoft came along; and by the time Microsoft had become the ‘evil  
 empire’ in the late 1990s, Larry and Sergey were starting Google. Was Google+  
 too clunky for you? No matter, try Facebook or Twitter. But as Moore’s law  
 runs out of steam, the dominant ﬁrms we have now may remain dominant for  
 some time – just like the railways dominated the second half of the nineteenth  
 century and the ﬁrst third of the twentieth. And there are many other sectors  
 where technology has enabled some players to lock in market dominance; as I  
 write in 2020, Amazon is the world’s most valuable company. We need to refresh  
 our thinking on antitrust law. There are some signs that this is happening [1044].  
 What would you hope the law to look like twenty years from now? How should  
 the safety, security and antitrust pieces ﬁt together?

**28.6** **Summary**

In the old days, the big question in a security engineering project was how you  
 know when you’re done. All sorts of evaluation and assurance methodologies  
 were devised to help. Now the world is different. We’re never done, and nobody  
 who says they are done should be trusted.

Security evaluation and assurance schemes grew up in a number of different

ecosystems. The US military spawned the original Orange Book, and inspired  
 both the FIPS 140 standards for cryptographic modules and the Common Cri-  
 teria, both of which attempted to spread the gospel of trustworthy systems to  
 businesses and to other countries. Safety certiﬁcation schemes evolved sepa-  
 rately in a number of industries – healthcare, aerospace and road vehicles to  
 name just three. Vendors game these systems all the time, and work to capture  
 the regulators where this is possible. Now that everything’s acquiring connec-

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 947 | Ross Anderson |

*28.6. SUMMARY*

tivity, you can’t have safety without security, and these ecosystems are merging.

In both safety and security, the emphasis will move from pre-market testing

to monitoring and response, which will include updating both devices already  
 in the ﬁeld and the services that support them. This will move beyond software  
 lifecycle standards towards the goal of a learning system that can recover quickly  
 even from novel hazards and attacks.

Things are improving, slowly. Back in the 20th century, many vendors never

got information security right. By 2010, the better ones were getting it more  
 or less right at the third or fourth attempt. In the future, everyone will be  
 expected to ﬁx their products reasonably promptly when they break, and to do  
 so for a reasonable period of time.

But the cost of all this, the entanglement of security with safety in all sorts

of devices and services, and their interaction with issues from discrimination  
 to globalisation and trade conﬂict, will make these issues increasingly the stuff  
 of global politics. The safety and security costs inﬂicted on us by tech, in its  
 broadest sense, will be in increasing tension with national ideas of sovereignty  
 and, at a more practical level, people’s ability to achieve by collective action  
 those goals that cannot be achieved through individual action or market forces.  
 Just as security economics was a hot topic in the 2000s and security psychology  
 in the 2010s, I expect that the politics of security will be a growth topic in the  
 2020s and beyond.

**Research problems**

In addition to the grand challenge of sustainable security I discuss in sec-  
 tion 28.5.2 above, there are many other open problems around assurance. We  
 really don’t know how to do assurance in complex ecosystems such as where  
 cars talk to online services and mobile phone apps. A second bundle of prob-  
 lems comes from the fact that as the worlds of safety and security are slowly  
 coming together, like a couple of galaxies slowly merging, we ﬁnd that safety  
 engineers and security engineers don’t speak each others’ languages, have incom-  
 patible sets of standards and even incompatible approaches to standardisation.  
 Working this out in one industry after another will take years.

Another big opportunity may be for lightweight mechanisms to improve real

deployed systems. Too many researchers take the view that ‘If it’s not perfect,  
 it’s no good.’ We have large communities of academics writing papers about  
 provable security, formal methods and about obscure attacks that aren’t found  
 in the wild because they don’t scale. We have large numbers of real problems  
 arising from companies corner-cutting on development. If programmers are

going to steal as much code as they can from stackexchange, do we need a  
 public-interest effort to clean up the examples there to get rid of the buffer  
 overﬂows? And do we have any chance of setting security usability standards  
 for tools such as crypto libraries and device permissions, so that (for example)  
 libraries that default to ECB would be forcibly retired, just like MD5 and SHA1?

Yet another is likely to be the testing of AI/ML systems, both before deploy-

ment and for continuous assessment. We already know, for example, that deep

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 948 | Ross Anderson |

*28.6. SUMMARY*

neural networks and other ML mechanisms inhale prejudice along with their  
 training data; because machine-vision systems are mostly trained on photos of  
 white people, they are uniformly worse at spotting people with darker skin,  
 leading to the concern that autonomous vehicles could be more likely to kill  
 black pedestrians [2026]. What will a learning system look like when it touches  
 a contentious social issue? How do you do continuous safety in a world where  
 not all lives are valued equally? How do we ensure that the security, privacy  
 and safety engineering decisions that ﬁrms take are open to public scrutiny and  
 legal challenge?

**Further Reading**

There’s a whole industry devoted to promoting the security and safety assurance  
 business, supported by mountains of your tax dollars. Their enthusiasm can  
 even have the ﬂavour of religion. Unfortunately, there are nowhere near enough  
 people writing heresy.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 949 | Ross Anderson |