**Chapter 22**

**Phones**

**I rarely had to resort to a technical attack. Companies can spend**

**millions of dollars toward technological protections and that’s**

**wasted if somebody can basically call someone on the telephone**

**and either convince them to do something on the computer**

**that lowers the computer’s defenses or reveals**

**the information they were seeking.**

– KEVIN MITNICK

**Privacy is not about hiding – privacy is about**

**human growth and agency.**

– CHRISTOPHER WYLIE

The protection of phones, the app ecosystem they support and the telecom-

munications networks on which they rely, is central to the modern world. First,  
 in the decade after the launch of the iPhone, the world moved from accessing  
 the Internet via PCs or laptops to using smartphones instead, and added bil-  
 lions of new users too. Whole business sectors are being revolutionised as they  
 move to apps; of the 5.5bn adults on earth, 5bn have phones, and 4bn of them  
 have smartphones. Second, the new generation of connected devices, from smart  
 speakers to cars, are very much like phones, often using the same platforms and  
 sharing the same vulnerabilities. Third, phones now provide the bedrock for  
 authentication: if you forget your password, you get an SMS to recover it – so  
 someone who can steal an SMS from you may be able to spend your money.  
 Fourth, mobile networks are critical to other infrastructure: electricity compa-  
 nies rely on mobile phones to direct their engineers when repairing faults, so if  
 the phone system goes down a few hours after the power does, there’s a real  
 problem. Finally, there’s public policy. While smartphones have revolutionised  
 the lives of the third-world poor by giving access to services such as banking,  
 they also facilitate surveillance and control.

The phone ecosystem is mind-numbingly complex, and to master it the se-

curity engineer needs not just general security knowledge such as crypto and  
 access controls, and knowledge of speciﬁc platforms such as Android and iOS,  
 but of mobile and ﬁxed-line networks too. The history of telecomms security

668

*22.1. ATTACKS ON PHONE NETWORKS*

is instructive. Early attacks were carried out on phone companies by enthusi-  
 asts (‘phone phreaks’) to get free calls; then the phone system’s vulnerabilities  
 were exploited by crooks to evade police wiretapping; then premium rate calls  
 were introduced, which brought in large-scale fraud; then when telecomms mar-  
 kets were liberalized, some phone companies started conducting attacks on each  
 other’s customers; and some phone companies have even attacked each other.  
 At each stage the defensive measures undertaken tended to be inadequate for  
 various reasons. The same cycle of exploitation then repeated with the Internet  
 – amateur hackers followed by debates about wiretaps followed by fraud and  
 tussles between companies and users; and as the two came together we’ve seen  
 lots of complex interactions. Now we see rapidly growing phone-based fraud  
 against banking systems, bad apps stealing people’s personal information and  
 high policy debates on the national security implications of 5G infrastructure.  
 How is the security engineer to navigate this?

The security of the phone as a platform depends on a number of things,

which I’ll deal with under two main headings.

1. First, there’s whether the network to which it’s attached has somehow

been compromised, whether by some kind of wiretap or by a SIM swap  
 attack which undermines the phone’s network identity.

2. Second, there’s the question of whether the device itself has been com-

promised, whether by malware rooting the operating system, or by the  
 installation of a potentially hostile application or library.

Phone security used to be all about the ﬁrst of these, but by now it’s mostly

about the second.

**22.1** **Attacks on phone networks**

The abuse of communications goes back centuries. Before Sir Rowland Hill

invented the postage stamp, postage was paid by the recipient. Unsolicited

mail became a huge problem – especially for famous people – so recipients were  
 allowed to inspect a letter and reject it rather than paying for it. People soon  
 worked out schemes to send short messages on the covers of letters which their  
 correspondents rejected. Regulations were brought in to stop this, but were  
 never really effective [1460].

A second set of abuses developed with the telegraph. Early optical tele-

graphs worked using semaphores or heliographs; people would bribe operators,  
 or ‘hack the local loop’ by observing the last heliograph station through a tele-  
 scope, to learn which horse had won before the local bookmaker did. Here too,  
 attempts to legislate the problem away were a failure [1818]. The problems got  
 worse when the electric telegraph brought costs down; the greater volumes of  
 communication, and the greater ﬂexibility that got built into and on top of the  
 service, led to more complexity and more abuse.

The telephone was to be no different.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 669 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

**22.1.1** **Attacks on phone-call metering**

Early phone-call metering systems were open to creative abuse.

*•* In the 1950’s, the operator in some systems had to listen for the sound of  
 so people practised hitting the coinbox with a piece of metal that struck  
 the right note.

*•* Initially, the operator had no way of knowing which phone a call had come  
 someone else – who would then be charged. Operators started calling back  
 to verify the number for international calls, so people worked out social  
 engineering attacks (‘This is IBM here, we’d like to book a call to San  
 Francisco and because of the time difference can our Managing Director  
 take it at home tonight? His number’s xxx-yyyy’). So payphone lines had  
 a warning to alert the operator. But the UK implementation had a bug: a  
 customer who had called the operator from a payphone could depress the  
 rest brieﬂy, whereupon he’d be reconnected (often to different operator),  
 with no warning this time that the call was from a payphone. He could  
 then call anywhere and bill it to any local number.

*•* Early systems also signalled the entry of a coin by one or more pulses, each  
 brief open circuit. At a number of colleges, enterprising students installed  
 ‘magic buttons’ which could simulate this in a callbox in the student union  
 so people could phone for free. (The bill in this case went to the student  
 union, for which the magic button was not quite so amusing.)

Attacks on toll metering have continued for over a century now. Most coun-

tries moved their payphones from coins to chip cards in the 1990s to cut the  
 costs of coin collection and vandalism, but as I remarked in section 18.5, the  
 design was often poor at ﬁrst and villains sold lots of bogus phone cards until  
 it got ﬁxed.

Other attacks involve what’s called *clip-on*: physically attaching a phone to

someone else’s line to steal their service. In the 1970s through the 1990s, when  
 international phone calls were very expensive, some foreign students would clip a  
 phone on to a residential line in order to call home, and the unsuspecting home  
 owner could get a huge bill. The Norwegian phone company had customer

premises equipment authenticate itself to the exchange before a dial tone was  
 given [994].

The UK phone company was not as enlightened as its Norwegian counter-

part, and had a policy of denying that wiretaps were possible, so it could just  
 collect the call charges from victim households. This occasionally caused col-  
 lateral damage, as a family in Cramlington was to ﬁnd out. The ﬁrst sign they  
 had of trouble was hearing a conversation on their line. The next was a visit  
 from the police who said there’d been complaints of nuisance phone calls. The  
 complainants were three ladies, all of whom had a number one digit different  
 from a number to which this family had supposedly made a huge number of

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 670 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

calls. When the family’s bill was examined, there were also calls to clusters of  
 numbers that turned out to be payphones; these had started quite suddenly at  
 the same time as the nuisance calls. When the family had complained later to  
 the phone company about a fault, their connection was rerouted and this had  
 solved the problem.

A report from the phone company’s maintenance engineer noted that the

family’s line had been tampered with at the distribution cabinet, but this was  
 against doctrine and the company later claimed the report was in error. It

turned out that a drug dealer had lived close by, and it seemed a reasonable  
 inference that he’d tapped their line in order to call his couriers at the payphones.  
 By using an innocent family’s phone line instead of his own, he not only saved  
 on the phone bill, but also had a better chance of evading police surveillance.  
 But both the police and the local phone company refused to go into the house  
 where the dealer had lived, claiming it was too dangerous – even though the  
 dealer had by now got six years in jail. The Norwegian phone company declined  
 an invitation to testify about clip-on for the defence. The upshot was that

the subscriber was convicted of making harassing phone calls, in a case widely  
 believed to have been a miscarriage of justice.

Stealing dial tone from cordless phones was another variant on the theme.

In the 1990s, this became so widespread in Paris that France Telecom broke  
 with phone company tradition and announced that it was happening, claim-  
 ing that the victims were using illegally imported cordless phones which were  
 easy to spoof [1097]. That was a bit cheeky, as most equipment seems to sim-  
 ply send a handset serial number to the base station rather than using the  
 DECT security mechanisms, which use cryptography patented by the French  
 company Alcatel. These mechanisms were proprietary but turned out to have  
 multiple weaknesses, as Erik Tews documented in 2012 after reverse engineering  
 them [1871]. DECT authentication is based on a weak block cipher; conﬁden-  
 tiality uses a weak stream cipher (a slightly more complicated version of A5/1  
 which I describe below in section 22.2.1) which can be broken with typically 234

effort; there are weak random number generators; while protocol failures include  
 a man-in-the middle attack, and a replay attack where you make a silent call  
 to collect keystream to decrypt a call you recorded earlier. It’s said that the  
 German intelligence services used DECT to train recruits in signal collection  
 and cryptanalysis. Since Tews’ work was published, the DECT standards body  
 suggests using AES instead but it’s not clear how many vendors can be both-  
 ered. The takeaway is that a cordless phone gives you no security against a  
 capable opponent nearby, and as the standard emerged during the Crypto Wars  
 of the 1990s you should have expected nothing else. As for clip-on fraud, it has  
 largely disappeared since services like Skype and WhatsApp made long-distance  
 calls cheap.

Social engineering gives another way in. A crook calls you pretending to

be from AT&T and asks whether you made a large number of calls to Peru on  
 your calling card. When you deny this, they say that, in order to reverse out  
 the charges, can you conﬁrm that your card number is 123-456-7890-6543? No,  
 you say (if you’re not really alert), it’s 123-456-7890-5678. Now 123-456-7890 is  
 your phone number and 5678 your password, so that crook can now bill calls to  
 you.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 671 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

Premium-rate phone services grew rapidly during the 1990s, leading scam-

sters to develop all sorts of tricks to get people to call them: pager messages, job  
 ads, fake emergency messages about relatives, ‘low cost’ calling cards with 0900  
 access numbers, you name it. Indeed the business of tricking people into calling  
 premium numbers enabled crooks to hone the techniques they now use in phish-  
 ing attacks. The 809 area code for the Caribbean used to be a favourite cover  
 for crooks targeting US subscribers; many people weren’t aware that ‘domestic’  
 numbers (numbers within the USA’s +1 international direct dialling code) in-  
 clude countries other than the relatively cheap USA and Canada. Even though  
 many people have now learned that +1 809 is ‘foreign’ and more expensive, the  
 introduction of still more Caribbean area codes, such as +1 345 for the Cayman  
 Islands, has made it even harder to spot such scams.

Phone companies advised their customers ‘Do not return calls to unfamiliar

telephone numbers’ – but how practical is that? Just as banks now train their  
 customers to click on links in marketing emails and thus make them vulnerable  
 to phishing attacks, so I’ve had junk marketing calls from my phone company  
 – even though I’m on the do-not-call list. Governments typically set up weak  
 regulators who avoid trying to regulate premium-rate operators, claiming it’s  
 too hard; and from time to time it all blows up. In the late 2000s, all the major  
 UK TV companies (including the state-owned BBC) ended up getting ﬁned for  
 getting viewers to phone in and vote, in all sorts of shows. Many of these are  
 recorded, so the calls were futile [1323]. Phone scams by broadcast stations  
 have been a recurring problem worldwide since radio broadcasting took off in  
 the 1920s, and got worse when TV went mainstream in the 1950s [2050]. It’s  
 also a recurring pattern that the biggest scams are often run by ‘respectable’  
 companies rather than by Russian gangsters.

**22.1.2** **Attacks on signaling**

The term ‘phone phreaking’ refers to attacks on signaling as well as pure toll  
 fraud. Until the 1980s, phone companies used signalling systems that worked  
 *in-band* by sending tone pulses in the same circuit that carried the speech. The  
 ﬁrst attack I’ve heard of dates back to 1952, and by the mid-to-late 1960s many  
 enthusiasts in both America and Britain had worked out ways of rerouting  
 calls. One of the pioneers, Joe Engresia, had perfect pitch and discovered as  
 a child that he could make free phone calls by whistling a tone he’d heard in  
 the background of a long-distance call. His less gifted colleagues used home-  
 made tone generators, of which the most common were called *blue boxes*. The  
 trick was to call an 0800 number and then send a 2600Hz tone that would *clear*  
 *down* the line at the far end – that is, disconnect the called party while leaving  
 the caller with a trunk line connected to the exchange. The caller could now  
 enter the number he really wanted and be connected without paying. Phone  
 phreaking was one of the roots of the computer hacker culture that took root in  
 the Bay Area and was formative in the development and evolution of personal  
 computers [1222]. Steve Jobs and Steve Wozniak ﬁrst built blue boxes before  
 they diversiﬁed into computers [722].

Phone phreaking started out with a strong ideological element. In those days

most phone companies were monopolies – large, faceless and unresponsive. In

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 672 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

America, AT&T was such an abusive monopoly that the courts eventually broke  
 it up; most phone companies in Europe were government departments. People  
 whose domestic phone lines had been involved in a service theft found they were  
 stuck with the charges. If the young man who had courted your daughter was  
 (unknown to you) a phone phreak who hadn’t paid for the calls he made to her,  
 you would suddenly ﬁnd the company trying to extort either the young man’s  
 name or a payment. Phone companies were also aligned with state security.  
 Phone phreaks in many countries discovered signalling codes or switch features  
 that would enable the police or the spooks to tap your phone from the comfort  
 of their desks, without having to send out a lineman to clip on a wiretap. Back  
 in the days of the Vietnam war and student protests, this was inﬂammatory  
 stuff. Phone phreaks were counterculture heroes, while phone companies were  
 hand-in-hand with the forces of darkness.

As there was no way to stop blue-box attacks so long as telephone signalling

was carried in-band, the phone companies spent years and many billions of  
 dollars moving to a signaling system called SS7 which is out-of-band, in effect on  
 a private Internet to which normal subscribers had no easy access. Gradually,  
 region by region, the world was closed off to blue-box attacks. This forced

attackers to become insiders.

**22.1.3** **Attacks on switching and conﬁguration**

Once telephone exchange switches became programmable, a second wave of  
 attacks targeted the computers. Typically these were Unix machines on a LAN  
 in the exchange, which also had machines with administrative functions such as  
 scheduling maintenance. By hacking one of these less well guarded machines,  
 a phreak could go across the LAN and break into the switching equipment –  
 or into other secondary systems such as subscriber databases. For a survey of  
 PacBell’s experience of this, see [388]; for Bellcore’s, see [1059].

Using these techniques, unlisted phone numbers could be found, calls could

be forwarded without a subscriber’s knowledge, and all sorts of mischief became  
 possible. A Californian phone phreak called Kevin Poulsen got root access

to many of PacBel’s switches and other systems in 1985–88: this apparently  
 involved burglary as much as hacking (he was eventually convicted of conspiring  
 to possess ﬁfteen or more counterfeit, unauthorized and stolen access devices).  
 He did petty things like obtaining unlisted phone numbers for celebrities and  
 winning a Porsche from Los Angeles radio station KIIS-FM. Each week KIIS  
 would give a Porsche to the 102nd caller, so Poulsen and his accomplices blocked  
 out all calls to the radio station’s 25 phone lines save their own, made the 102nd  
 call and collected the Porsche. He was also accused of unlawful wiretapping and  
 espionage; these charges were dismissed. In fact, the FBI came down on him  
 so heavily that there were allegations of an improper relationship between the  
 agency and the phone companies, along the lines of ‘you scratch our backs with  
 wiretaps when needed, and we’ll investigate your hacker problems’ [690].

The FBI’s sensitivity does highlight the fact that attacks on phone company

computers are used by foreign intelligence agencies to conduct remote wiretaps.  
 Some of the attacks mentioned in [388] were from overseas, and the possibility  
 that such tricks might be used to crash the whole phone system in the context

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 673 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

of an information warfare attack worried the NSA [727, 1106]. Countries that  
 import their telephone exchanges rather than building their own just have to  
 assume that their telephone switchgear has vulnerabilities known to the sup-  
 plier’s government. (During the invasion of Afghanistan in 2001, Kabul had two  
 exchanges: an old electromechanical one and a new electronic one. The USAF  
 bombed only the ﬁrst.)

Many real attacks involved insiders, who misconﬁgured systems to provide

free calls through special numbers. This didn’t matter much when the phone  
 company’s marginal cost of servicing an extra phone call was zero, but with  
 the proliferation of value-added services in the 1990s, and with deregulation  
 giving rise to cash payments between phone companies, it got serious [460]. In  
 a hack reminiscent of Poulsen, two staff at British Telecom were dismissed after  
 they each won ten tickets for Concorde from a phone-in offer at which only one  
 randomly selected call in a thousand was supposed to get through [1914].

As for outsiders, the other ‘arch-hacker’ apart from Poulsen was Kevin Mit-

nick, who got arrested and convicted following a series of break-ins which made  
 him too the target of an FBI manhunt. They initially thought he was a foreign  
 agent who was abusing the US phone system to wiretap sensitive US targets. As  
 I mentioned in Chapter 3, he testiﬁed after his release from prison that almost  
 all of his exploits had involved social engineering. He wrote a book on deception  
 that became a classic [1325]. In congressional testimony, he came up with the  
 quote at the head of this chapter: “Companies can spend millions of dollars  
 toward technological protections and that’s wasted if somebody can basically  
 call someone on the telephone and either convince them to do something on the  
 computer that lowers the computer’s defenses or reveals the information they  
 were seeking”. Phone companies, like other ﬁrms, are vulnerable to careless

insiders as well as malicious insiders.

Fast forward to 2020, and one worrying development is the growth of switch-

ing exploits. A number of telcos now give SS7 access to corporate customers,  
 for example if they want to send bulk SMS messages to authenticate customers.  
 Access to the switch fabric lets them play the kind of games that Poulsen and  
 Mitnick got up to in the 1980s. For example, if I want to hack your Gmail  
 account, I send a message to your mobile service provider saying that you’ve  
 roamed into my network. I then start an account recovery at Google, which  
 sends an SMS to reset your password. As I noted in sections 3.4.1 and 12.7.4,  
 this is now in active use for bank fraud; the ﬁrst instance of its use to steal  
 money from bank customers was in Germany in 2016, when they were moved  
 without their knowledge to another network; there was a similar fraud in Lon-  
 don in 2019 [489]. SS7 has also been abused by Saudi Arabian MNOs to track  
 Saudi dissidents in the USA [1054]. Most major telcos in developed countries  
 now use some SS7 ﬁrewalling, and allow or deny remote access depending on  
 their roaming agreements. If there is such an agreement, a ﬁrm given SS7 access  
 by the remote telco can either steal a phone to get its SMS messages, or get it  
 to do premium fraud. Forensics can be hard if there’s a complaint from a single  
 user; the best you can do may be to look for roaming charges. If there are a  
 thousand cases the bank might be motivated to go to the operator. But banks  
 and their bulk SMS contractors are paying operators for SS7 access, opening up  
 the formerly closed system. In short, we used to think that attacks involving

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 674 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

SS7 were the preserve of nation states, but that is no longer the case.

**22.1.4** **Insecure end systems**

The next major vulnerabilities of modern phone networks were insecure terminal  
 equipment and feature interaction.

There have been many exploits of voicemail, whether implemented as an

answering machine on customer premises or, as common now, a cloud service.  
 Exploits start with tricking someone into calling a premium-rate number, and  
 escalate to journalists and others hacking voicemail via the default PINs that  
 many people don’t bother to change. The most notorious case was the murder,  
 on the 21st of March 2002, of the English schoolgirl Millie Dowler. In 2011 it  
 transpired that an investigator working for the News of the World, then the  
 UK ﬂagship of the Murdoch empire, had hacked Millie’s voicemail, interfered  
 with the police investigation in the process, and may have caused some of her  
 messages to be deleted, giving Millie’s family a false hope that she might still be  
 alive. The resulting outrage led to the closure of the newspaper, several criminal  
 convictions – including the imprisonment in 2014 of David Cameron’s publicist  
 Andy Coulson, a former News of the World editor – and a public inquiry into  
 press standards.

But the really big frauds that exploit insecure end systems tend to target

companies and government departments, as they have the ability to pay big  
 phone bills. Attacks on corporate *private branch exchange* systems (PBXes)  
 had become big business by the mid-1990’s and cost business billions of dollars  
 a year [467]. PBXes are usually supplied with facilities for *reﬁling* calls, also  
 known as *direct inward system access* (DISA). The company’s sales force could  
 call in to an 0800 number, enter a PIN or password, and then call out again  
 taking advantage of the low rates a large company can get for long-distance calls.  
 As you’d expect, these PINs become known and get traded by villains [1352].  
 The result is known as *dial-through* fraud.

In many cases, the PINs are set to a default by the manufacturer, and never

changed by the customer. Many PBX designs also have ﬁxed engineering pass-  
 words that allow remote maintenance access, and prudent people reckon that  
 any PBX will have at least one back door to give easy access to law enforcement  
 and intelligence agencies (it’s said, as a condition of export licensing). Such  
 features get discovered and abused. In one case, the PBX at Scotland Yard was  
 compromised and used by criminals to reﬁle calls, costing the Yard a million  
 pounds, for which they sued their telephone installer. The crooks were never  
 caught [1868]. One of the criminals’ motivations is to get access to communi-  
 cations that will not be tapped. Businesses who’re the victims of such crimes  
 ﬁnd the police reluctant to investigate, and the phone companies aren’t helpful  
 – they don’t like having their bills disputed [1624].

In a notorious case, Chinese gangsters involved in labour market racketeer-

ing – smuggling illegal immigrants from Fujian, China, into Britain – hacked the  
 PBX of an English district council and used it to reﬁle over a million pounds’  
 worth of calls to China. The gang was tackled by the police after a number of  
 its labourers died; they were picking shellﬁsh in Morecambe Bay when the tide

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 675 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

came in and drowned them. The council had by now discovered the discrep-  
 ancy in its phone bills and sued the phone company for its money back. The  
 phone company argued that it wasn’t to blame, even though it had supplied  
 the insecure PBX. Here, too, the gangsters were interested not just in saving  
 money but in evading surveillance. (Indeed, they routed their calls to China via  
 a compromised PBX in Albania, so the cross-border segment of the call, which  
 is most likely to be monitored by the agencies, was between numbers their col-  
 lection systems wouldn’t touch; the same trick seems to have been used in the  
 Scotland Yard case, where the crooks made their calls via the USA.)

Such cases apart, dial-through fraud is mostly driven by premium rate ser-

vices and the crooks are in cahoots with premium line operators. Most compa-  
 nies don’t understand the need to guard their ‘dial tone’ and don’t know how to  
 even if they wanted to. PBXes are typically run by company telecomms man-  
 agers who know little about security, while the security manager often knows  
 little about phones. This is changing as company phone networks adopt VOIP  
 technologies and merge with the data network. Estimates of the losses from PBX  
 fraud sustained by business worldwide fell from $4.96bn in 2011 to $3.88bn in  
 2017, with about half the latter ﬁgure now VOIP rather than classical PBX [91].

Exploits of insecure end-systems affect domestic subscribers too. Premium-

rate mobile malware arrived in 2006, when the Red Browser worm cashed out by  
 sending $5 SMSs to Russia [941]; this scaled up after Android came along, and  
 we’ll discuss mobile malware in section 22.3.1.4. And now that phones are used  
 more and more for tasks such as voting, securing entry into apartment buildings,  
 checking that offenders are observing their parole terms, and authenticating  
 ﬁnancial transactions, more motives are created for ever more creative kinds  
 of mischief, and especially for hacks that defeat caller-line ID. Since the early  
 2000s, there have been warnings that caller-line ID hacks, SMS spooﬁng and  
 attacks on the SS7 signaling could be used for fraud. This is now reality, and  
 we’ll discuss it in more detail later in this chapter.

**22.1.5** **Feature interaction**

Phone manipulation often involves feature interaction.

*•* Inmates at the Clallam Bay Correctional Center in Washington state, who  
 system which the phone company (‘Fone America’) introduced to handle  
 collect calls automatically. The system would call the dialled number and a  
 synthesised voice would say: “If you will accept a collect call from...(name  
 of caller)...please press the number 3 on your telephone twice.” Prisoners  
 were supposed to state their name for the machine to record and insert.  
 The system had, as an additional feature, the ability to have the greeting  
 delivered in Spanish. Inmates did so, and when asked to identify them-  
 selves, said “If you want to hear this message in English, press 33.” This  
 worked often enough that they could get through to corporate PBXes  
 and talk the operator into giving them an outside line. The University of  
 Washington was hit several times by this scam [696].

*•* Many directory-enquiry services will connect you to the number they’ve

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 676 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

just given you, as a premium service for motorists who can’t dial while  
 driving. It can also be used to defeat mechanisms that rely on endpoint  
 identiﬁcation. Naughty children use it to call sex lines despite call barring,  
 while naughty grown-ups use it to prevent their spouses seeing lovers’  
 numbers on the family phone bill [1456].

*•* Call forwarding is a source of many scams. In the old days, it was used  
 forward their teacher’s calls to a sex line. Nowadays, it can be both

professional and nasty. For example, a fraudster may tell a victim to

conﬁrm their phone number with the bank by dialing a sequence of digits  
 – which forwards incoming calls to a number controlled by the attacker.  
 So the bank’s callback mechanisms are defeated.

*•* Conference calls can be exploited in all sorts of ways. For example, foot-  
 them to be at home during a match, and to prove this by calling the pro-  
 bation service, which veriﬁes their caller ID. So you get your partner to  
 set up a conference call with the probation service and your mobile. If the  
 probation officer asks about the crowd noise, you tell him it’s the TV and  
 you can’t turn it down or your mates will kill you. (And if he wants to  
 call you back, you get your partner to forward the call.)

**22.1.6** **VOIP**

In *voice over IP* (VOIP), voice traffic is digitised, compressed and routed over the  
 Internet. This had experimental beginnings in the 1970s; products started ap-  
 pearing in the 1990s, and it became big business from the mid-2000s. Nowadays,  
 most traditional phone calls are digitized and sent over IP networks belonging  
 to the phone companies, so in a technical sense almost all phone calls are now  
 ‘VOIP’. But though my home phone pretends to be a plain old telephone, my  
 lab phone is now a born-VOIP device that offers conference calling and all sorts  
 of other complicated features that I don’t understand.

The most popular VOIP protocol, the Session Initiation Protocol (SIP), has

had its share of vulnerabilities [2069] but is mostly attacked through poor con-  
 ﬁgurations, for which many actors are constantly scanning; a PBX can get over  
 a million messages a day trying to register as an extension, and then attempt-  
 ing to call high-cost numbers in less developed countries [1271]. As I noted in  
 section 22.1.4, the VOIP segment of frauds against corporate PBX systems was  
 about $2bn a year by 2017 [91]. The broader interaction with security is compli-  
 cated. Corporate security policies can result in ﬁrewalls refusing to pass VOIP  
 traffic. The current political tussle is over robocalls, which can hide caller ID  
 more easily if they go over VOIP. The FCC voted in 2020 to insist that telcos  
 implement by the end of June 2021 a suite of protocols, STIR/SHAKEN, which  
 authenticate callers over SIP [326]. Another regulatory issue is that govern-  
 ments want emergency calls made through VOIP services to work reliably, and  
 provide information about the location of the caller. But an IP packet stream  
 can come from anywhere, and no-one owns enough of the Internet to guarantee  
 quality of service. And although a VOIP handset looks like a phone and works

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 677 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

like a phone, if the power goes off, so does your service. Then you’re forced to  
 fall back on the mobile network. So now it’s the mobile network rather than the  
 traditional one that is the default emergency system.

**22.1.7** **Frauds by phone companies**

Phone fraud is not just a story of crooked customers committing toll fraud  
 against telcos, and defrauding other customers by exploiting mechanisms that  
 the telcos have no real incentive to harden, There are many scams by unscrupu-  
 lous telcos. The classic scam is *cramming*, where a rogue phone company bills  
 lots of small sums to unwitting users. Billing was designed in the days when  
 phone companies were monopolies, usually state-owned, and assumes that phone  
 companies trust each other: if company A creates a *call data record* (CDR) say-  
 ing that a customer of telco B called their subscriber, they just pass it on to  
 telco B, which pays up. (It has no incentive to quibble, as it gets a cut.)

I was myself the victim of an attempt at cramming. On holiday in Barcelona,

my wife’s bag was snatched, so we called up and cancelled the phone that  
 she’d had in it. Several months later, we got a demand to pay a few tens of  
 dollars roaming charges recently incurred by that SIM card in Spain. In all  
 probability, the Spanish phone company was simply cramming a few charges to  
 a number that they’d seen previously, in the knowledge that they’d usually get  
 away with it. My wife’s former MNO insisted that even though she’d cancelled  
 the number, she was still liable for calls billed to it months afterwards and had  
 to pay up. We got out of the charges only because I’d met the company’s

CEO at an academic seminar and was able to get his private office to ﬁx the  
 problem. Customers without such access usually get the short end of the stick.  
 Indeed, UK phone companies’ response to complaints has been to offer customers  
 ‘insurance’ against fraudulent charges. That they can get away with this is a  
 clear regulatory failure. There are many variants: if you call an 800 number  
 in the USA, the company may say “Can we call you right back?” and if you  
 agree then you’re deemed to have accepted the charges, which can be at a high  
 premium rate. The same can happen if you respond to voice prompts as the  
 call progresses.

Another problem is *slamming* – the unauthorized change of a subscriber’s

service provider without their consent. It would be a mistake to assume that  
 cramming and slamming are just done by small ﬂy-by-night operators. AT&T  
 was one of the worst offenders, having been ﬁned not only for slamming, but for  
 forging signatures of subscribers to make it look as if they had agreed to switch  
 to their service. They got caught when they forged a signature of the deceased  
 spouse of a subscriber in Texas.

Yet another is the exploitation of international calls for premium-rate scams.

The abuse of domestic premium-rate numbers led regulators in many countries  
 to force phone companies to offer premium-rate number blocking to subscribers.  
 The telcos got round this by disguising premium rate numbers as international  
 ones. I mentioned scams with Caribbean numbers in section 22.1.1, and many  
 other phone companies from small countries got into the act. Such scams bene-  
 ﬁt from an international agreement (the Nairobi Convention) that stops phone  
 companies selectively blocking international destinations. Advisories from gov-

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 678 | Ross Anderson |

*22.1. ATTACKS ON PHONE NETWORKS*

ernments still warn of ‘wangiri’ scams where you get a call that rings once, in  
 the hope you’ll call back – to an international premium number. However these  
 seem to have stopped; an extensive study of robocalls in 2020 found no evidence  
 of them any more [1543]. There are many reasons why scams may be moving  
 away from the telco platform to the app ecosystem; the interaction between  
 scams and regulation is complex.

By the time smartphones came along, the phone companies had got used

to taking a cut of high-value service delivery, ranging from parking meters in  
 London to ferry tickets in Finland. As malware became widespread on mobile  
 phones, the botnet herders who control subverted phones could pay for goods  
 and services by SMS. Many new services were made possible by the smartphone  
 revolution and payment moved from SMS to payments via apps. SMS abuses  
 have got to the point that neither Google nor Apple allows normal apps to send  
 or receive text messages. We might pause to think of the industry’s economics.  
 Why have telcos never felt a duty of care towards their customers?

**22.1.8** **Security economics of telecomms**

Phone and cable companies have extremely high ﬁxed costs and very low marginal  
 costs. Building a nationwide network costs billions and yet the cost of handling  
 an additional phone call or movie download is essentially zero. As I discussed  
 in the chapter on Economics, this has a couple of implications.

First, there’s a tendency towards dominant-ﬁrm markets. For many years

telephone service was considered in most countries to be a ‘natural monopoly’  
 and operated by the government; the main exception was the USA where the old  
 AT&T system was heavily regulated. After the breakup of AT&T following an  
 antitrust case, and Margaret Thatcher’s privatisation of BT in the UK, the world  
 moved to a different model, of regulated competition. The details vary from one  
 country to another but, in general, some sectors (such as mobile phones) had  
 a ﬁxed number of allowed competitors; others (such as long-distance provision)  
 were free for companies to compete in; and others (such as local loop provision)  
 remained monopolies but were regulated.

Second, the competitive sectors (such as long-distance calling) saw prices

drop quickly to near zero. Some sectors were made competitive by apps: Skype  
 and WhatsApp made international calls essentially free.

In many telecomms markets, the outcome is *confusion pricing* – products

are continually churned, with new offerings giving generous introductory dis-  
 counts to compete with the low-cost providers, but with rates sneakily raised  
 afterwards. There is constant bundling of broadband access with mobile service  
 and TV offerings. If you can be bothered to continually check prices, you can  
 get good deals, but often at the cost of indifferent service. If you don’t have the  
 time to keep scrutinising your broadband and mobile phone bills, you can get  
 some unpleasant surprises.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 679 | Ross Anderson |

*22.2. GOING MOBILE*

**22.2** **Going mobile**

Since their beginnings as an expensive luxury in 1981, mobile phones have be-  
 come one of the big technological success stories. By 2020, we now have over  
 ﬁve billion subscribers; it’s said that over a billion phones were sold in 2019  
 alone. In developed countries, most people have at least one mobile, and many  
 new electronic services are being built on top of them. Growth has been rapid  
 in developing countries too, where the wireline network is often dilapidated and  
 people used to wait years for phone service. In many places it’s the arrival of  
 mobile networks that connected villages to the world. This has brought many  
 beneﬁts, and new crimes too. Both developed steadily as the technology was  
 evolved and deployed.

Mobile phone security has developed as the abuse has. The ﬁrst generation

of mobile phones (1G) used analog signals and the handset simply sent its serial  
 numbers in clear over the air link1. So villains built devices to capture these  
 numbers from calls in the neighborhood, or reprogrammed phones to steal ID  
 from other phones nearby. One of the main customers was the *call-sell opera-*  
 *tion* that would steal phone service and resell it cheaply, often to immigrants  
 or students who wanted to call home. The call-sell operators would hang out  
 at known pitches with cloned mobiles, and their customers would queue up to  
 phone home for a few dollars. The call-sell market was complemented by the  
 criminal market for anonymous communications: people hacked mobile phones  
 to use a different identity for each call. Known as *tumblers*, these were particu-  
 larly hard for the police to track [944]. 1G phones did not encrypt voice traffic,  
 so anyone could casually eavesdrop on calls with a radio receiver, yet despite  
 this the possibility of caller anonymity led to their use in crime. The demand for  
 serial numbers grew rapidly and satisfying it was increasingly difficult, even by  
 snooping at places like airports where lots of mobiles get turned on. So prices  
 rose, and as well as passive listening, active methods started to get used.

Mobile phones are cellular: the operator divides the service area up into

cells, each covered by a base station. The mobile uses the base station with the  
 strongest signal, and there are protocols for handing off calls from one cell to  
 another as the customer moves about. Early active attacks consisted of a fake  
 base station, typically at a place with a lot of passing traffic such as a freeway  
 bridge. As phones passed by, they heard a stronger signal and attempted to  
 register by sending their serial numbers and passwords.

Various mechanisms were tried to cut the volume of fraud. Most oper-

ators ran intrusion-detection systems to watch out for suspicious patterns of  
 activity, such as too-rapid movement or a rapid increase in call volume or du-  
 ration. Vodafone also used RF ﬁngerprinting, a military technology in which  
 signal characteristics arising from manufacturing variability in the handset’s  
 radio transmitter are used to identify individual devices and tie them to the  
 claimed serial numbers [776].

1In the US system, there were two of them: one for the equipment, and one for the

subscriber.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 680 | Ross Anderson |

*22.2. GOING MOBILE*

**22.2.1** **GSM**

The second generation of mobile phones (2G) adopted digital technology. The  
 *Global System for Mobile Communications* (GSM) was founded when 15 com-  
 panies signed up to the GSM Association in 1987 and secured political support  
 from the EU; service was launched in 1992. The designers of GSM set out to  
 secure the system against cloning and other attacks: their goal was that GSM  
 should be at least as secure as the wireline system. What they did, how they  
 succeeded and where they failed, make an interesting case history.

The industry initially tried to keep secret the cryptographic and other pro-

tection mechanisms which form the core of the GSM protocols. This didn’t

work: some eventually leaked and the rest were discovered by reverse engineer-  
 ing. I’ll describe them brieﬂy here. Mobile networks consist of a *radio access*  
 *network* (RAN) and a *core network* (CN), and each mobile network has two  
 databases, a *home location register* (HLR) that contains the location of its own  
 mobiles, and a *visitor location register* (VLR) for the location of mobiles which  
 have roamed in from other networks. These databases enable incoming calls to  
 be forwarded to the correct cell.

The handsets are commodity items, personalised using a *subscriber identity*

*module* (SIM) – a smartcard you get when you sign up for a network service,  
 and which you load into your handset. The SIM can be thought of as containing  
 three numbers:

1. there may be a personal identiﬁcation number that you use to unlock the

card;

2. there’s an *international mobile subscriber identiﬁcation* (IMSI), a unique

number that maps on to your mobile phone number;

3. ﬁnally there is a *subscriber authentication key Ki*, a 128-bit number that

serves to authenticate that IMSI and is known to your home network.

There is also a handset serial number, the *international mobile equipment*

*identiﬁcation* (IMEI). The protocol used to authenticate the handset to the  
 network runs as follows (see Figure 22.1). On power-up, the SIM emits the  
 IMSI, which the handset sends to the nearest base station along with the IMEI.  
 The IMSI is relayed to the subscriber’s HLR, which generates ﬁve *triplets*. Each  
 triplet consists of:

*•* RAND, a random challenge;

*•* SRES, a response; and

*• Kc*, a ciphering key.

The algorithm is that RAND is encrypted under the SIM’s authentication

key *Ki*, giving SRES concatenated with *Kc*:

*{RAND}Ki* = (*SRES|Kc*)

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 681 | Ross Anderson |

*22.2. GOING MOBILE*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SIM� | Mobile� | |  |  | | --- | --- | |  |  | |  | | | HLR� |
| BSC� | VLR� |

Figure 22.1: – GSM authentication system components

The encryption method is up to the issuer; an early standard called Comp128

turned out to be insecure [1971, 1972], so issuers nowadays use hash functions  
 or constructions using AES.

Anyway, the triplets are sent to the *base station controller* (BSC), which

now presents the ﬁrst RAND to the mobile. It passes this to the SIM, which  
 computes SRES. The mobile returns this to the base station and if it’s correct  
 the mobile and the base station can now communicate using the ciphering key  
 *Kc*. So the whole authentication protocol runs as in Figure 22.2.

SIM *!* HLR IMSI  
 HLR *!* BSC (RAND, SRES, *Kc*), ...  
 BSC *!* SIM RAND  
 SIM *!* BSC SRES  
 BSC *!* mobile *{*traffic*}Kc*

Figure 22.2 – GSM authentication protocol

There are several vulnerabilities in this protocol. First, the base station isn’t

authenticated, so it’s easy for a wiretapper to use a false base station to intercept  
 calls. Such devices, known as *IMSI catchers* in Europe and *StingRays* in the  
 USA, are now standard law-enforcement equipment2. Second, in most countries  
 the communications between base stations and the VLR pass unencrypted on  
 microwave links3. This allows bulk interception by intelligence agencies, and in  
 many cases access to the triples needed to spoof or decrypt traffic.

The introduction of GSM caused signiﬁcant shifts in patterns of crime. The

authentication mechanisms made phone cloning difficult, so the villains switched  
 to buying phones using stolen credit cards, using stolen identities or bribing in-  
 siders [2034]. Robbery was the next issue, with a spate of media stories about  
 kids being mugged for their phones. Mobile phone crime did indeed increase  
 190% between 1995 and 2002, but to keep this in context, the number of sub-  
 scribers went up 600% in the same period [865]. Some of the theft is bullying –  
 kids taking smaller kids’ phones; some is insurance fraud by subscribers who’ve  
 dropped their phones in the toilet and report them as stolen as their insurance  
 doesn’t cover accidental damage; but there is a hard core of theft where muggers  
 take phones and sell them to fences. Many of the fences either work at mobile  
 phone shops that have authorised access to tools for reprogramming the IMEI,

2When 2G was designed, a base station ﬁlled a whole room and cost $100k, so it might have

seemed reasonable to ignore man-in-the-middle attacks. Nowadays all it takes is a low-cost  
 software radio.

3The equipment can encrypt traffic, but the average phone company has no incentive to

switch the cryptography on.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 682 | Ross Anderson |

*22.2. GOING MOBILE*

the serial number in the handset, or else have links to organised criminals who  
 ship the handsets abroad4.

Prepaid mobile phones appeared from about 1997, enabling the industry to

expand rapidly to people without credit ratings, including both poor people  
 in rich countries and everyone in poor countries. By 2008, prepaids made up  
 90% of the market in Mexico but 15% in the USA. During the 2010s, billions  
 of people got access not just to calls and texts but to online information and  
 payment services.

Prepaid phones also made anonymous communication practical. The issues

include not just evading police wiretapping but fraud, stalking, extortion, bul-  
 lying and other kinds of harassment. However, prepaid phones only protect you  
 from the police if they don’t try very hard. Most criminals don’t have any clue  
 of the level of operational discipline needed to stop traffic analysis. As I already  
 remarked, one alleged 9/11 mastermind was caught when he used a prepaid  
 SIM from the same batch as one that had been used by another Al-Qaida mem-  
 ber; and after the failed 21/7 London bombings, one would-be bomber ﬂed to  
 Rome, where he was promptly caught. He had changed the SIM in his mobile  
 phone en route; but call records show not just the IMSI from the SIM, but  
 also the IMEI from the handset. If you’ve got all the world’s police after you,  
 just changing the SIM isn’t anything like enough. Operational security requires  
 some understanding of how networks operate.

In addition to authentication, 2G was supposed to provide two further kinds

of protection – location security and call content conﬁdentiality.

The location security mechanism is that when a mobile is registered to a

network, it is issued with a *temporary mobile subscriber identiﬁcation* (TMSI),  
 which acts as its address in that network. This is a lightweight mechanism; it  
 is defeated trivially by IMSI catchers, which pretend to be a base station in a  
 different network.

2G GSM also provides some call content conﬁdentiality by encrypting the

traffic between the handset and the base station once authentication and reg-  
 istration are completed. The speech is digitized, compressed and chopped into  
 packets; each packet is encrypted by xor-ing it with a pseudorandom sequence  
 generated from the ciphering key *Kc* and the packet number. The algorithm  
 commonly used in Europe is A5/1. This is a stream cipher that, like Comp128,  
 was originally secret; like Comp128, it was leaked and attacks were quickly  
 found on it [248]. By the mid-2000s, law enforcement suppliers were selling

|  |
| --- |
| devices that would break the key in under a second, enabling a surveillance  team to hoover up all the GSM tra�c and decrypt it, so they could then pick  out conversations of interest. Phones also supported an even weaker algorithm called A5/2, which was licensed for export to non-EU countries5 and which can  be broken almost instantly. As I mentioned above in section 22.1.1, the DECT  standard for cordless phones is somewhat similar, and also weak. The embassies  of major powers round the world have roof structures that indicate antennas for  capturing local telephone tra�c, and the Snowden papers conﬁrm that the NSA |

4In recent smartphone designs, the IMEI is supposed to be unalterable; some Android

phones keep it in TrustZone.

5There was a row when it emerged that Australia was using A5/2.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 683 | Ross Anderson |

*22.2. GOING MOBILE*

collects local phone traffic at US diplomatic missions.

In addition to passive bulk collection, targeted active collection can exploit

protocol tricks.

GSM vendors introduced a third cipher, A5/3, which is based on a strong

block cipher known as Kasumi and became standard in third-generation mobile  
 phones. But there’s the *bidding-down attack* which exploits the fact that the  
 initial algorithm negotiation is in plaintext. The IMSI catcher simply tells the  
 handset to use a weaker cipher. Elad Barkan, Eli Biham and Nathan Keller  
 realised that this can be done retrospectively [171]. If you’re following a suspect  
 who uses his mobile, you record the call, including the initial protocol exchange  
 of challenge and response. Once he’s ﬁnished, you switch on your IMSI-catcher  
 and cause him to register with your bogus base station. The IMSI-catcher tells  
 his phone to use A5/2 rather than A5/1, and a key is duly set up – with the  
 IMSI-catcher sending the challenge that was used before. So the mobile phone  
 generates the same key *Kc* as before. As this is now being used in a weak cipher,  
 it can be cracked quickly, giving access to the conversation already recorded.  
 A5/2 has now been retired; handsets that cannot use A5/1 or A5/3 communicate  
 in plaintext. However A5/1 is easy to break with modern equipment.

Phone companies, equipment vendors and ISPs are now compelled to pro-

vide for local law-enforcement access, but other countries often want access  
 too and the wiretap facilities are often so poorly engineered that they can be  
 abused [1707]. In 2004-5, persons unknown (but presumed to be from the NSA  
 or CIA) tapped the mobile phones of the Greek Prime Minister and about a  
 hundred of that country’s political, law enforcement and military elite during  
 the Athens Olympics, by subverting the wiretapping facilities built into Voda-  
 fone’s Greek network. Both Vodafone, and their equipment supplier Ericsson,  
 were heavily ﬁned [1550]. Colleagues and I warned about this problem years  
 ago [4] and the Snowden disclosures suggest that it has got steadily worse. I’ll  
 discuss it at greater length in Part III.

Anyway, the net effect is while the 2G GSM security mechanisms were de-

signed to provide slightly better protection than the wireline network in coun-  
 tries allowed to use A5/1, and somewhat worse protection elsewhere, they now  
 provide slightly worse protection everywhere because of the range of exploits  
 that can be industrialised by third parties.

**22.2.2** **3G**

The third generation of digital mobile phones was initially known as the *Uni-*  
 *versal Mobile Telecommunications System* (UMTS) and now as the *Third Gen-*  
 *eration Partnership Project* (3gpp, or just 3G). The acronym 3gpp is still used  
 for the standards body working on 4G, 5G and beyond. 3G entered service  
 in 2003–2004 and is due to be retired in 2022, after which mobile devices that  
 cannot use 4G or 5G are supposed to fall back to 2G. This may happen mostly  
 in sparsely-populated rural areas where it is uneconomic to install the newer  
 4G and 5G technologies and the far greater backhaul transmission they need.  
 3G uses spread-spectrum technology on the radio access network, and instead  
 of the 9.6kb/s of standard 2G and the tens of kilobits per second of the 2.5G

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 684 | Ross Anderson |

*22.2. GOING MOBILE*

variant (GPRS), 3G data rates are in the hundreds of thousands of bits per sec-  
 ond. 3G’s vision was to enable all sorts of mobile services, from mobile TV to  
 laptops that just go online anywhere. It laid the foundation for the smartphone  
 revolution.

The overall security strategy is described in [1976], and the security archi-

tecture is at [1961]. The crypto algorithms A5/1 and A5/2 are replaced by A3,  
 based on a block cipher called Kasumi [1021], which in turn is based on a design  
 by Mitsuru Matsui called Misty, which has now withstood public scrutiny for  
 two decades [1245]. All keys are now 128 bits. Cryptography is used to protect  
 the integrity and conﬁdentiality of both message content and signalling data,  
 rather than just content conﬁdentiality, and the protection runs from the hand-  
 set to the core network, rather than simply to the local base station. So picking  
 up the keys, or the plaintext, from the base station or microwave backhaul is no  
 longer an attack. The authentication is now two-way rather than one-way. The  
 theory was that this would end the vulnerability to rogue base stations, and  
 IMSI catchers wouldn’t work any more. In practice, they work ﬁne as they just  
 tell the target handset to fall back to 2G operation. 3G also has also a proper  
 interface for local interception [1962].

In the basic 3G *authentication and key agreement* (AKA) protocol, the au-

thentication runs from the handset to the visitor location register. The home  
 location register is now known as the *home environment* (HE) and the SIM as  
 the *UMTS SIM* (USIM). The home environment chooses a random challenge  
 RAND as before and enciphers it with the USIM authentication key *Ki* to gen-  
 erate a response RES, a conﬁdentiality key *CK*, and integrity key *IK*, and an  
 anonymity key *AK*.

*{RAND}K* = (*RES|CK|IK|AK*)

There is also a sequence number SEQ known to the HE and the USIM. A

MAC is computed on RAND and SEQ, and then the sequence number is masked  
 by exclusive-or’ing it with the anonymity key. The challenge, the expected

response, the conﬁdentiality key, the integrity key, and the masked sequence  
 number made up into an *authentication vector AV* which is sent from the HE  
 to the VLR. The VLR then sends the USIM the challenge, the masked sequence  
 number and the MAC; the USIM computes the response and the keys, unmasks  
 the sequence number, veriﬁes the MAC, and if it’s correct returns the response  
 to the VLR.

|  |  |
| --- | --- |
| USIM *!* HE  HE *!* VLR  VLR *!* USIM  USIM *!* VLR | IMSI (this can optionally be encrypted)  RAND, XRES, *CK*, *IK*, *SEQ � AK*, *MAC*  RAND, *SEQ � AK*, *MAC* |

Fig 20.4 – 3gpp authentication protocol

The 3G standards set out many other features, including identity and lo-

cation privacy mechanisms, backwards compatibility with 2G, mechanisms for  
 encrypting authentication vectors in transit from HEs to VLRs, and negotiation  
 of various optional cryptographic mechanisms.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 685 | Ross Anderson |

*22.2. GOING MOBILE*

As with 2G, its design goal was that security should be comparable with

that of the wired network [922] and the net effect was a modest improvement:  
 bulk eavesdropping on the air link is prevented by higher-quality mechanisms,  
 although targeted attacks by IMSI catchers still work by exploiting fallback. In  
 a number of countries, third-generation mobiles were hard for the police to tap  
 in the ﬁrst few years, as they had to integrate their systems with those of the  
 network operators to operate at any scale greater than tactically.

**22.2.3** **4G**

Fourth-generation mobile networks were ﬁrst rolled out in 2009, and accounted  
 for most mobile subscriptions (4.2bn of the 8bn) by 2019 [981]. They use IP  
 throughout, unlike 2G and 3G which had circuit-switched core networks. The  
 radio access network changed from 3G’s spread spectrum to *frequency-domain*  
 *equalization* (FDE) schemes, making very high bit rates possible despite multi-  
 path radio propagation (echoes). The higher data rates made apps such as

Google Maps and Snapchat work much better, and made video streaming apps  
 possible. There is actually a family of standards that has evolved during the  
 2010s, supporting bandwidths in the megabits up to tens of megabits per second.  
 The 4G security standards rowed back from 3G by limiting encryption to the  
 link between the handset and the base station, though to be fair most apps now  
 encrypt data at the application layer. The authentication and key agreement  
 (AKA) protocol is very similar to 3G, although the nomenclature has changed.  
 The handset is now the UE or user equipment while the HE/HLR is now the  
 *home subscriber server* (HSS). The base station functionality is split into an  
 Evolved NodeB (eNodeB) base station and a smaller number of Mobility Man-  
 agement Entities (MMEs), which handle the AKA exchange, make admission  
 decisions, supply session keys to the base stations and handle law enforcement  
 access. The idea was that the MMEs can be housed in protected spaces or at  
 least made tamper-resistant (people talked about TPMs but no operator seems  
 to have implemented them).

The three main weaknesses in 4G are that local traffic at a base station

(or MME) can still be monitored by anyone who can take it over; that the  
 user equipment’s identity is sent to the network in the clear, or masked using  
 a *Globally Unique Temporary Identity* (GUTI) which is fairly weak, like its  
 predecessor the TMSI [918]; and that the home network delegates authentication  
 to the serving network [362]. SS7 is replaced by a control protocol suite called  
 Diameter, where messages can be optionally encrypted, but as the operators  
 trust each other it’s vulnerable to many of the same types of attack [426]. It  
 started off with fewer abusable functions, but they got put back in following  
 business pressure.

*Rich Communications Services* (RCS) became widely available during 2019

thanks to support from Google in its Messages app. It is intended to replace  
 SMS with richer chat features including geolocation exchange, social presence  
 information and voice-over-IP. Also known as SMS+, +Message or joyn, it pro-  
 vides many of the same services as WhatsApp, but without the end-to-end  
 encryption, as it’s a telco hosted product. Many of the initial implementations  
 are insecure as the telcos haven’t conﬁgured them correctly [1696].

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 686 | Ross Anderson |

*22.2. GOING MOBILE*

For decades, phone security has been kept weak at the behest of the secu-

rity and intelligence community. Yet this strategy blew back when it turned  
 out that Russian agents in the USA compromised the communications of FBI  
 counterintelligence agents who used push-to-talk cellphones [579]. We haven’t  
 been told whether they were 3G or 4G, or what the speciﬁc exploits were, but it  
 was so bad that in December 2016 the Obama administration kicked out three  
 dozen Russian diplomats. They had also been obsessed with getting premises  
 with line of sight to the CIA HQ at Langley, Virginia.

**22.2.4** **5G and beyond**

Fifth-generation networks entered service in 2019, promising a further signiﬁcant  
 improvement over 4G in terms of bandwidth and latency. The main driver at  
 present is bandwidth; mobile traffic grew by 68% between Q3 2018 and Q3  
 2019, mostly from video, and growth at over 25% is anticipated up till 2025, by  
 which time almost half the traffic worldwide will be 5G [981]. Again, there’s  
 an evolving family of standards, with complexity increasing still further. Initial  
 deployments use *non-standalone mode* (NSA) which reuse the 4G control plane  
 (and even the 4G towers) but boost the data rate. The real excitement is about  
 *standalone mode* (SA) which will follow. 5G makes it cheaper and easier for  
 mobile network operators to build new capacity, not just at existing frequencies,  
 but at millimeter-wave frequencies over 20GHz, which will mean much larger  
 numbers of small base stations on lamp posts, bus stops and so on (this will  
 also limit the time available to do authentication handshakes). Network energy  
 efficiency and area traffic capacity could be up two orders of magnitude, while  
 connection density, mobility and data rates could go up one order. Availability  
 is a high priority; after the 2016 Brussels bombings, the police couldn’t get  
 network service on their phones because of congestion, and had to ﬁnd wiﬁ  
 hotspots to talk to each other.

The terminology changes yet again. Each tiny base station is now a *dis-*

*tributed unit* (DU) and is controlled by a *centralised unit* (CU), which is also in  
 the ﬁeld but counted as part of the core network. The encryption goes from your  
 device to the CU, and from there it’s protected using IPSec to the *access man-*  
 *agement function* (AMF), which replaces the MME boxes. The authentication  
 and key agreement protocols are much the same (XRES is renamed HXRES).  
 One material improvement is that your device identity is sent to your home net-  
 work encrypted under its public key, so location privacy will be harder to break;  
 and we’re told that IMSI catchers won’t work any more6. Passive and active  
 attacks by fake base stations seem still possible, including man-in-the-middle  
 attacks that downgrade a device to a previous generation of technology, and  
 could be used to deplete the batteries of energy-critical devices [1712].

However the whole core network moves to the cloud, including all the law-

enforcement access mechanisms. Instead of defending familiar technologies, mo-  
 bile network operators will depend on new ones that they don’t understand and  
 which most will just buy from the cheapest vendor. One mistake in conﬁg-

uration, and things could be world readable; and unless something like SGX

6We heard that before with 3G: the wiretappers just forced fallback to 2G. We hear that

the intelligence agencies are lobbying to break this, in alliance with the big data carriers.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 687 | Ross Anderson |

*22.2. GOING MOBILE*

can be made to work, the cloud providers’ governments may well be able to  
 get access by serving warrants on them rather than on the operators. The use  
 of SDN in the core cloud network opens up still more questions, of which the  
 most troublesome long-term may be whether 5G becomes an end-run round net  
 neutrality, enabling network operators to customise offerings to each application  
 by performance (and price). Meanwhile the speciﬁcations are complex and the  
 implementations are still ﬂaky. As the standards evolve, one ﬁght is between  
 the big data carriers who want to manipulate traffic to break net neutrality and  
 claw their way up the value chain, versus the big mobile network operators who  
 want end-to-end trust. In theory traffic edits will be signed by the ﬁrm that  
 does the editing, but nobody seems to know how that will work. Another is  
 that the US government is trying to prevent Huawei getting a critical mass of  
 installations outside China; the 2019 annual report of the UK National Cyber  
 Security Centre (part of GCHQ) noted that signiﬁcant supply-chain risks have  
 developed over 2010–19, for which market drivers were insufficient to ensure an  
 adequate response [1393]. In 2020, with anti-Chinese sentiment rising with the  
 coronavirus pandemic and the end of ‘one country two systems’ in Hong Kong,  
 the UK government decided to ban Huawei from selling 5G network equipment  
 from the end of 2020 and remove its existing equipment by 2027. A longer-  
 term resolution may depend on a third tussle, between the ‘bellheads’ and the  
 ‘netheads’: between ﬁrms like Nokia and Huawei who take a phone-industry ap-  
 proach and culture, and insurgents such as Rakuten whose culture is from the  
 computer industry and which will happily virtualise everything in sight once it’s  
 in the cloud [609].

What about 6G and 7G? Telecomms researchers talk about the former see-

ing evolution in the radio access network to support a diversity of apps with  
 different requirements for peak bandwidth, latency, service quality and power  
 consumption [1454]; and the latter having thousands of micro-satellites to de-  
 ploy 200Mbps broadband over all the earth’s surface. The arrival of stream-  
 ing games, augmented reality and (perhaps) autonomous vehicles will create  
 demand for ultra-low-latency cloud services, so rather than having our data  
 shipped off to a few dozen data centres run by Google, Facebook, Microsoft and  
 Amazon, we may see edge clouds with clusters of servers in each town, perhaps  
 even in the buildings that used to house the old telephone exchanges. Then,  
 just as the dotcom boom in the late 1990s forced us to partition web services  
 into the active processes at the core and the rest that could be served more or  
 less statically and thus cached locally in CDNs, we’ll have to host some of the  
 active stuff locally too.

**22.2.5** **General MNO failings**

Regardless of the generation of radio link technology in use, there are some  
 common failings of MNOs whose root causes lie in the economics and regu-  
 lation of the industry. One is the rapidly growing attacks on authentication  
 functions supported by mobile phones. In addition to the SS7 security issues  
 we discussed in section 22.1.3, which apply also to wireline telcos, the mobile  
 world has brought us SIM swapping, channel jacking and the theft of cookies  
 from authenticator apps. Many of these have security economics at their root:

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 688 | Ross Anderson |

*22.2. GOING MOBILE*

there is some misalignment of incentives between the various principals in the  
 system.

In section 3.4.1 we introduced SIM swap attacks, where the attacker per-

suades the victim’s telco to issue a new SIM card on the victim’s account. This  
 can open the door to all sorts of mayhem; individuals can have their lives trashed  
 by attackers who take over their online accounts. Celebrities are targets: in Au-  
 gust 2019, Twitter CEO Jack Dorsey had his account taken over for an hour  
 and used to send racist and antisemitic tweets, causing commentators to wonder  
 whether someone who took over President Trump’s twitter account might start  
 World War 3 [1340]. As I mentioned in section 12.7.4, SIM-swap attacks are  
 mostly used in 2020 against the customers of banks and bitcoin exchanges, and  
 often involve phone company insiders. Yet the response of phone companies has  
 been at best patchy. The only major US MNO making SIM swapping harder is  
 Verizon [712]. But not all countermeasures help all users: if they are optional,  
 then the company can more easily disclaim losses by the customers who don’t  
 opt to use them. The ﬁrst MNO to take action was MTN in South Africa in  
 2003, which enabled users to designate a second SIM to authorise SIM replace-  
 ment; curiously, this was the phone company involved in the ﬁrst SIM-swap  
 fraud case in 2007, which I described in section 12.7.4. Phone companies can  
 also help relying parties detect SIM swaps by sending a hash of the IMSI as a  
 response to the second-factor SMS; but few do so. We discussed the often ad-  
 versarial attitude of phone companies toward their customers in section 22.1.8;  
 MNOs are no different in this respect from legacy wireline phone companies.  
 Indeed, they may be worse because most of their customers in most countries  
 are prepayment customers.

Another example of MNOs and their suppliers feeling unable to do customer

security properly is SIMjacking. In 2013, Karsten Nohl warned that many SIMs  
 in use were easy to hijack, because of features built in to facilitate over-the air  
 software update. The industry retorted that it wasn’t a problem as SIM cards  
 could run only signed software [1582]. In 2019, it emerged that governments had  
 been using this for surveillance [1107]. MNOs’ relationship with their customers  
 has always been somewhat adversarial, and they are compelled in many coun-  
 tries to run middleperson attacks on demand. When a suspect’s mobile phone  
 browser visits an unencrypted URL, the MNO serves police malware instead.  
 Such network injection attacks can be done tactically, with IMSI-catchers, but  
 doing them at the MNO is more convenient. This practice started in less devel-  
 oped countries but has now spread as far as Germany [1443]. We will discuss  
 government surveillance, and the tensions it has generated with security since  
 the crypto wars, in section 26.2.7.3.

The real underlying problem for the MNOs is that they lost control of ser-

vices. For various reasons, they were unable to engage with developers and

promote an app ecosystem from which they could extract value. They ended up  
 being commoditised – bit shifters who have to maintain the infrastructure, but  
 who see the monopoly proﬁts they used to enjoy being creamed off by others.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 689 | Ross Anderson |

*22.3. PLATFORM SECURITY*

**22.3** **Platform security**

The second part of the phone story is the app ecosystems. These ﬁx some

problems, and create others: the most acute security problem is whether the  
 platform itself is trustworthy, or whether your phone might act against your  
 interests. This has been a growing concern since programmable phones came  
 along in the early 2000s. For the back story see the second edition of my

book which describes the state of play in 2007. Brieﬂy, before the iPhone came  
 along, security was fragmented along the supply chain, with chip designers, chip  
 makers, OS vendors, handset OEMs and MNOs passing the buck while they  
 tussled over DRM and over control. MNOs refused to allow OEMs to have any  
 relationship with the customer. As I remarked in the chapter on Access Control,  
 Arm launched TrustZone in 2004; by 2007, several hundred viruses and worms  
 were being detected in Symbian phones each year, and vendors responded with  
 access controls, code signing, and so on.

Apple changed the world in several ways at once. First, it broke the taboo on

OEMs having a relationship with the customer. Second, it made it much easier  
 for third party vendors to write apps. Third, it made the App Store central to a  
 platform strategy, which it monetised by taking a share of both music downloads  
 and software. This entailed a semi-closed platform. Devices could go online  
 either through an MNO or via wiﬁ, and could switch easily between the two as  
 needed. The effect was to shift power from the MNO to Apple. Google launched  
 Android the following year, with a strategy of making a similar platform as open  
 as possible7, allowing anyone to write apps for Android phones. They aimed  
 to provide a minimum level of trust, to enable the ecosystem to grow. They  
 remembered that Microsoft had grabbed most of the PC software market from  
 Apple in the early 1980s by offering a more open platform that got the network  
 effects going in their favour and hoped to do the same with phones, leaving the  
 iPhone as a niche product for the rich. This did not in the end happen, and  
 we now have two large ecosystems that have converged in a number of ways.  
 But Apple’s monetisation strategy does give it a better incentive to maintain  
 its platforms, and iPhones are typically patched for at least ﬁve years while  
 Android products are patched for three, and often less.

Both the iPhone and Android launched with security architectures I describe

in the chapter on Access Control; both approaches aim to separate apps from  
 each other and to prevent them from subverting the platform itself. The main  
 processor is not the whole story, as phones contain dozens of other CPUs, and  
 there have been vulnerabilities discovered in DSPs too which can affect handsets  
 from multiple OEMs [1212]. I also discussed in the chapter on Side Channels  
 how a bad app could, for example, use the phone’s accelerometer and gyro to  
 work out a password or PIN being entered into another app, even if denied  
 direct access to the screen. The combination of rich sensors and a huge range  
 of applications makes security and privacy services at the platform level rather  
 complex. Both the Android and iPhone security mechanisms have been reﬁned  
 over time, with more controls added to block or mitigate the more ﬂagrant  
 abuses. However they can best be understood as an ecosystem, rather than as

7subject to the regulators’ insistence that the baseband software which controls the device’s

RF behaviour had to be locked down

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 690 | Ross Anderson |

*22.3. PLATFORM SECURITY*

a list of protection options.

This ecosystem is truly immense. By 2019, 56% of all Internet access globally

was from mobile devices, but 63% in the USA and 80% in India [1252]. It

consists at the very least of the apps that run on the two families of mobile  
 devices themselves, and the back-end services they rely on. The boundaries

are hard to deﬁne. We probably have to include the ad ecosystems that app  
 developers bundle with their products. Do we include the web services that  
 mobile devices access from browser apps? Do we include voice telephony, now  
 that this is migrating to apps like WhatsApp, Skype and Signal? What about  
 other devices, from watches to cars, that run mobile operating systems and  
 apps? It may be simplest to start with the app families.

**22.3.1** **The Android app ecosystem**

Android is the most widely deployed end-user operating system, found not just  
 in phones but in tablets, watches, TVs, cars and other devices – a total of over  
 2bn monthly active devices. Its platform security model is described by Ren´e  
 Mayrhofer and colleagues from Google in [1252], and in section 6.2.8 I discussed  
 the technical architecture. Actions are based on three-party consent: the user,  
 the developer and Google should all agree. The implementation is that rather  
 than giving a userid to the end user, as in a conventional \*nix system, Android  
 runs each app with a separate userid; data in private app directories is controlled  
 by the app, while data in shared storage is controlled by the end user, and there  
 are mandatory access control mechanisms to ensure that critical system data  
 remain under the control by the platform, unless it’s rooted. So long as this  
 does not happen, the user cannot be tricked into letting a bad app access or  
 overwrite the data of other apps. The threat model includes everything from  
 physical attacks and wiretapping through the exploitation of vulnerabilities in  
 the operating system, libraries and other apps; it’s assumed that users will be  
 tricked into installing malicious apps [1252]. Apps sold via Google’s Play store  
 are scanned for malware (though the scanning isn’t perfect).

However, Google takes 30% of revenues from sales of apps, and refuses to

host adult apps. This has driven many vendors of paid and adult apps to use less  
 secure distribution channels such as OEM deals, third-party stores and their own  
 websites [1823]. Since 2014 Google has offered to upload non-Play-store apps for  
 scanning when they’re ﬁrst run, but the risk of evil apps is ever present. Many  
 more apps are somewhat predatory, even if they’re distributed by apparently  
 respectable businesses such as hardware vendors, MNOs and security ﬁrms. The  
 sad fact is that user data has become a major commodity; little else might have  
 been expected given that most apps are free and the ecosystem is driven as  
 much by ad revenue as anything else. One major consequence is that Android  
 does not support the most critical permission for privacy – allowing the user to  
 control Internet access for an app. (Blackberry allowed users to deny Internet  
 access.) This pleases ad companies as otherwise many users would turn off

internet access for the ﬂashlight/game/compass app the moment they installed  
 it. If this displeases you, you can get ﬁrewall apps that pretend to be VPNs and  
 can block other apps’ access to the Internet. But of course most users go with  
 the default, of letting the ad ecosystems harvest just about everything.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 691 | Ross Anderson |

*22.3. PLATFORM SECURITY*

**22.3.1.1** **App markets and developers**

App markets mitigate some security problems while amplifying others. As the  
 Android ecosystem is open, anyone can be a developer and distribute the soft-  
 ware they write through the Play Store. This makes a huge market available to  
 novice developers, who can get simple apps running with little effort. The fact  
 you have to use the framework with the Android SDK constrains developers in  
 potentially useful ways. Although fragmentation greatly impedes the update  
 process for operating systems, app updates are easy if you use an app store that  
 pushes updates.

However the developer rapidly encounters both technical and business com-

plexity. Some simple apps are little more than a customised browser for an  
 online back end; others exercise a single feature of the phone in new ways, as  
 ﬂashlight apps do. But how uniform is that feature? How many versions of  
 Android do you need to support? Do you need to test on hundreds of different  
 handsets? There are now test frameworks to help, but fragmentation is a real  
 issue if your app uses the rich hardware features on many modern phones. For  
 example, people developing contact-tracing apps for coronavirus have struggled  
 with the variation in bluetooth performance between different handsets. An-  
 other example is where developers want to protect really sensitive information,  
 such as key material in banking apps. Arm hoped that developers would use  
 TrustZone but this turned out to be so hard given the variation between OEMs,  
 handsets and software versions, that most turned to obfuscation instead. An-  
 droid then provided KeyStore, which lets an app store its keys in TrustZone or a  
 Secure Element or other cryptoprocessor if available, and block other apps from  
 using them. Some developers prefer obfuscation in the hope of blocking mal-  
 ware that roots the phone and can thus pretend to be the app; as I mentioned  
 in section 12.7.4, some banking regulators insist on this.

Business complexity can come from the application itself, or from the ecosys-

tem’s underlying economics: platform companies, device vendors, app develop-  
 ers, app publishers (who add all sorts of ads), ad networks, toolsmiths and end  
 users all have different incentives. There are different rules for paid apps, apps  
 allowing in-app purchases and free apps. The rules for identifying users are com-  
 plex: the user’s consent is needed to use some UIDs (IMEI, IMSI, phone number  
 and ad ID) but not others such as MAC address and hardware ﬁngerprint.

**22.3.1.2** **Bad Android implementations**

The ﬁrst bundle of systemic security problems to become obvious as Android  
 became widespread around 2010 was the poor quality of the engineering work  
 by many of the OEMs who licensed it. One example was factory reset. There’s  
 a thriving trade in second-hand phones, as rich users buy the latest models  
 and their old phones end up being sold. You might think that when you do a  
 factory reset on your phone, that clears all your personal information, not just  
 from shared storage but from app storage as well. But it’s hard to get this right  
 because of all the interactions with how Flash memory is organised on a typical  
 phone; there may be an embedded multimedia card (eMMC) and virtual SD  
 card, with their own wear-levelling mechanisms. If the OEM’s engineers don’t

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 692 | Ross Anderson |

*22.3. PLATFORM SECURITY*

take the trouble to implement secure deletion, then the all-too-common outcome  
 is that someone who buys your phone second-hand can retrieve the Google  
 master cookie and access the Gmail account associated with the phone [1757].  
 For several years I bought Google’s own-brand Nexus and Pixel phones and  
 never sold them after use, but many people get phones subsidised by a contract  
 and locked to the MNO, which sells them in second-hand markets afterwards –  
 often in less developed countries. (It is prudent to assume that Android phones  
 in LDCs have been rooted and had remote access Trojans installed by local  
 distributors.)

These quality problems extend to TrustZone and its Trusted Execution En-

vironment (TEE), as implemented by various chipset vendors. For example,  
 Qualcomm’s TEE system lets a *trusted app* (TA) map in memory regions of the  
 host OS, and as a result any insecure TA can let an adversary root the device.  
 Other problems allow attacks on the TEEs of the other four vendors: the soft-  
 ware security mechanisms used in trusted environments lag the state of the art  
 by several years, with absent or weak ASLR, excessively large TCBs, informa-  
 tion leaks through debugging channels, no execution prevention, multiple side  
 channels and no good ways to revoke wicked or vulnerable TAs – of which there  
 are plenty. See David Cerdeira and colleagues for a survey of these issues [403].

However the biggest security problem with Android implementations is poor

after-sales support. Many OEMs only support the version that’s currently being  
 actively marketed; they are reluctant to spend engineer time backporting ﬁxes to  
 old versions. A 2015 survey revealed that 87% of active devices were insecure,  
 averaged over 2011–15, because they were running versions of the operating  
 system that contained known vulnerabilities. In many cases, the OEM simply  
 did not make ﬁxes available [1880]. This had already been identiﬁed as a problem  
 by Google by 2011; the company offered OEMs access to cut-price components  
 if they undertook to patch their systems, but this got little traction. Google  
 now offers certiﬁcation programs for both vendors and apps, but the problems  
 go deeper than just OEM engineering effort. If a vulnerability is found in, say,  
 the OpenSSL or Bouncy Castle cryptographic library, this ﬁx has to propagate  
 to Linux, then to Android, them to each OEM, and then in many cases to each  
 mobile network operator – as the MNOs control updates for phones that are  
 locked to the network. Each of these steps can take several months, and each  
 can be neglected for commercial reasons [1880]. This raises thorny issues around  
 coordinated disclosure, which we’ll discuss in section 27.5.7.2, and regulation,  
 which we’ll discuss in the last chapter of this book.

**22.3.1.3** **Permissions**

Consent has been a wicked problem from the beginning, as we noted in the  
 chapter on Access Control. In early versions of Android, an app’s manifest

speciﬁed the access rights it demanded and the user would have to approve  
 them all on installation in order to run it. This led to widespread abuse, as  
 most users would just click approval to get the installation done, and a lot of  
 utility apps became machines for harvesting and reselling your address book,  
 browser history and other personal data. Already in 2012, research showed that  
 only 17% of users paid attention during installation, and only 3% could answer

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 693 | Ross Anderson |

*22.3. PLATFORM SECURITY*

basic questions about what was going on [676]. In 2015, Android 6 moved to  
 the Apple model of approving access to such resources on ﬁrst use. Indeed,  
 progressive restrictions of the more dangerous permissions have driven platform  
 evolution more than anything else. Android 6 also made ﬁne-grained location  
 access a separate permission; Android 7 limited apps’ access to the metadata  
 of other apps; Android 8 randomised MAC addresses and mandated the use of  
 a single Advertising ID for monetisation; Android 9 limited access to sensors  
 when an app is in background mode and restricted access to the phone and call  
 logs; and Android 10 restricted location access in background mode.

Google now provides several dozen permissions, and developers have always

been able to deﬁne custom permissions when making services available to other  
 apps; thousands of these are deﬁned by hardware vendors, MNOs, security ﬁrms  
 and Internet browsers [741]. These further balkanise the ecosystem and make it  
 even harder for users (and developers) to understand.

An analysis of the consent problem by Yasemin Acar and colleagues breaks

it up into comprehension of permissions, and attention to permissions, by both  
 users and developers [10]. There are both usability and incentive failures on  
 both sides. It’s clear enough why a predatory ﬂashlight app wants access to  
 my address book; many failures are more subtle. Developers are just trying to  
 make stuff work so they can ship it, while users are just trying to access some  
 service or other. Developer usability is a signiﬁcant source of bugs; we’ve noted  
 this elsewhere (e.g. in section 5.5) but it looms larger in appiﬁed ecosystems as  
 the developers have to drive the application framework APIs to get useful work  
 done. A substantial minority of developers request more permission than they  
 need out of ignorance or confusion, and this holds even for system apps whose  
 developers should know better. Google failed to implement fail-safe defaults;  
 the APIs are confusing and poorly documented. This drove developers to copy  
 each others’ code via fora such as stackexchange, to an even greater extent than  
 with conventional development8.

**22.3.1.4** **Android malware**

As Android is an open platform, for which anyone can write apps, it has at-  
 tracted a lot of harmful software. As we mentioned in section 22.1.4, premium-  
 rate phone malware arrived in 2006 with the Red Browser worm; Android’s  
 arrival turned mobile malware from a niche activity into a mainstream prob-  
 lem. Deﬁnitions here are hard, as many apps are harmful in different ways to at  
 least some people; here I focus on apps that act secretly against the interests of  
 the user that installed them. I’ll discuss bad programs installed by OEMs and  
 MNOs later in section 22.3.1.6.

Malware can be bulk or targeted, and it can come from private-sector crim-

inals or state actors. Most of it by volume is of the bulk private-sector variety,  
 and most of that comes through regular distribution channels. As well as the  
 millions of apps in the Play Store, alternative markets are widely used, espe-  
 cially in countries like China and Iran where the Play Store is censored. The

8It also drove Acar and her colleagues to look at usability from the developers’ view-

point [11], creating an important new area of security research which I mentioned in the  
 research problems section at the end of the chapter on Access Control.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 694 | Ross Anderson |

*22.3. PLATFORM SECURITY*

largest single source of malware has been the Play Store, with a signiﬁcant mi-  
 nority of apps being harmful at some times, while some alternative markets have  
 on occasion removed most of their apps for being harmful. Apps may be born  
 harmful, or libraries on which they rely may become bad, or the bad guys may  
 buy failing app companies, just as they snap up domains of former banks. One  
 of the biggest crime rings exposed recently did hundreds of millions of dollars of  
 ad fraud by buying Android apps and using their user data to train bots that  
 then clicked on ads [1738]; such scams exploit other kinds of malware too. The  
 measurement problems are non-trivial, as over 60 anti-virus ﬁrms label apps us-  
 ing different criteria and classify them into different families. There are several  
 hundred families active at any one time.

A 2018 survey by Guillermo Suarez-Tanguil and Gianluca Stringhini anal-

ysed 1.2m samples collected over 2010–17, and classiﬁed them into over a thou-  
 sand families [1842]. Since 2012, most of them have involved repackaging, where  
 the malware dev takes a legitimate app (the *carrier*) and adds harmful code (the  
 *rider*). This is industrialised by repackaging many benign carriers with variants  
 of the same malicious rider. The riders may try to root the phone for persis-  
 tent access, and drop a remote access Trojan (RAT) that can earn money at  
 the direction of a command-and-control server, just as with regular PC mal-  
 ware. Monetisation strategies have evolved; in 2010 the focus was on making  
 premium-rate calls, but by 2018 it had shifted to ad fraud and the exﬁltration of  
 personal information. The great majority of riders use obfuscation tricks such  
 as encryption, while only a quarter of benign apps do this (Facebook’s app uses  
 obfuscation as a defence against user data and keys being stolen by malware,  
 particularly RATs in less developed countries). Riders are mostly native code  
 rather than Java (or Kotlin, which replaced it as the official Android language  
 of choice in 2019).

Banking Trojans stand out among the more targeted varieties of private-

sector malware. A common approach is the *overlay attack* where the malware  
 tricks the user into allowing it to use Android Accessibility Services, which  
 enables it to build an overlay over (for example) your banking app so it can  
 capture the screen and input data, under the control of a remote command  
 server [396]. Android malware has been stealing bank SMSes for some time,  
 and Google has pushed back by allowing only approved apps the permission  
 to read SMSes; the latest development in 2020 is that the Cerberus banking  
 malware can now steal Google authenticator cookies too [431].

States already used targeted malware in intelligence and law-enforcement

missions, and by 2012 vendors such as Gamma had produced mobile-phone  
 versions of their products that were found in multiple jurisdictions [1231]. Such  
 malware also seeks root access but implants spyware. Recent examples of bulk  
 malware deployment come from Turkey, which in 2018 was using man-in-the-  
 middle devices on the T¨urk Telekom network to deploy spyware [1218], and  
 China, which sets website traps for Uighurs’ phones [393]. Bulk state-actor

malware can include mandating doctored versions of apps in some jurisdictions;  
 Skype was available in China from 2005 only through a local distributor, Tom  
 Online, which repackaged it to scan for words forbidden by Chinese censors.  
 After Microsoft bought Skype, they took back control from 2013, but the app  
 was banned from app stores accessible in China from 2017 [1347].

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 695 | Ross Anderson |

*22.3. PLATFORM SECURITY*

There are technical abuses where apps defeat the permission framework while

stopping short of rooting your phone. Joel Reardon and colleagues ran 88,000  
 Android apps in an instrumented virtual environment to look for apps abus-  
 ing side channels [1588]. They found two large Chinese companies, Baidu and  
 Salmonads, using the SD card as a covert channel, so that ads which could read  
 the phone’s IMEI could store it for those which could not. They also found 42  
 apps getting the IMEI when they shouldn’t, using ioctl system calls, and over  
 12,000 with the code to do so.

**22.3.1.5** **Ads and third-party services**

Mobile phone apps typically incorporate third-party services to support ads,  
 social network integration and analytics for a range of purposes from crash re-  
 porting to A/B testing. Such services can track users across multiple apps, even  
 without their consent. An example of what can go wrong comes from Cam-  
 Scanner, an app downloaded by over 100m people for scanning and managing  
 documents. At some point, the app was updated to add a new advertising

network that contained a malicious module. Negative reviews led antivirus re-  
 searchers to take a look, and it turned out that the module was dropping Trojans  
 on to people’s phones [796].

Third-party services are a fairly opaque part of the ecosystem, as they are

not directly visible to the user. Some light has been shed by a survey carried  
 out by Abbas Razaghpanah and colleagues, using a VPN app used by 11,000  
 volunteers to monitor traffic to and from their phones [1586]. They mapped  
 over 2,000 *advertising and tracking services* (ATS), including hundreds that had  
 not previously been reported, and found that a substantial minority (39%) did  
 cross-device tracking; 17 of the top 20 had a presence on the web as well as in  
 the app ecosystem. Eight of the top ten reserved the right, in their privacy poli-  
 cies, to share data with other organisations. The largest of all were Alphabet  
 and Facebook, but ﬁrms whose whole business consists of ATS, such as Chart-  
 boost, Vungle and Adjust, have a signiﬁcant share and are relatively unknown  
 to users. App developers often use several such services simultaneously. Paid  
 apps have the fewest trackers, free apps have more, and free apps that allow  
 in-app purchases, often of premium services, tend to have the most.

Mutual trust issues are discussed by Yasemin Acar and colleagues [10]. App

developers have to trust ad networks, as they execute in the app sandbox and  
 inherit its permissions. Ad libraries exploit apps in various ways, such as loading  
 insecure code from web services and stealing users’ private information; app  
 developers return the compliment by stealing money from the networks with  
 fake click events, just like malware developers. (The boundaries are a bit fuzzy,  
 as they were before in the world of the PC; there’s predatory behaviour at just  
 about every layer of the stack.)

There are many examples of children’s apps collecting personal data with-

out parental consent, contrary to the US Children’s Online Privacy Protection  
 Act (COPPA): Irwin Reyes and colleagues scanned 5,855 of the most popular  
 free children’s apps and found that most of them potentially violated COPPA  
 because of the way they used third-party SDKs; these typically enable devel-  
 opers to disable third-party tracking and advertising but most developers don’t

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 696 | Ross Anderson |

*22.3. PLATFORM SECURITY*

bother. Worse, 19% of the apps were collecting personally identiﬁable informa-  
 tion using SDKs that banned this in children’s apps [1599]. This study led to  
 legal action by state attorneys general, which might encourage app developers  
 to take the law more seriously. There are other practices contrary to the EU  
 GDPR and its ePrivacy Directive, but EU regulators seem reluctant to get en-  
 gaged, as the ATS industry is overwhelmingly based in the USA, and amounts  
 to a substantial invisible export. Even from the viewpoint of the US authorities,  
 most of the ATS specialists don’t even have a COPPA policy, leaving regulatory  
 compliance to their customers.

Most people expect that if they pay for an app, they get more privacy. But

given that developers rely on third-party services for analytics as well as ads, this  
 costs effort, which many developers can’t be bothered to make. Catherine Han  
 and colleagues compared free and paid versions of the same app and found that  
 a third of the paid versions were just as predatory in terms of data collection;  
 another sixth collected at least some of the same data; three-quarters used the  
 same permissions; and almost all had the same security policy. Looking at

paid/free app pairs designed for families, she found that the majority of paid  
 apps violated COPPA in the same way as the free versions [859].

**22.3.1.6** **Pre-installed apps**

Julien Gamba and colleagues studied the ﬁrmware distributed by over 200 ven-  
 dors worldwide [741]. Distributions typically reﬂect a partnership between a  
 handset OEM and an MNO, with various affiliated developers, ad networks and  
 distributors. They can be poorly controlled; there have been multiple cases of  
 malware ﬁnding its way in, as well as software to do mass-scale data collection  
 for commercial or regulatory reasons. Some phones also have diagnostic or sup-  
 port modes that could be exploited by wicked apps. Most of the pre-installed  
 apps are not available in the Play Store and thus appear to fall outside the  
 conventional framework. Some are from ﬁrms like Facebook and AccuWeather  
 which are known to collect personal data aggressively; many of these are not the  
 public versions of these ﬁrms’ apps; and many pre-installed apps use mobile an-  
 alytics or targeted advertisement libraries. What’s more, 74% of the non-public  
 apps do not seem to get updated, and 41% remained unpatched for 5 years  
 or more [741]. Many have sensitive custom permissions in order to perform

such tasks as mobile device management for enterprise customers, call block-  
 ing, and VPN services. Behavioral analysis showed that a signiﬁcant proportion  
 of pre-installed apps could access and disseminate user and device identiﬁers,  
 conﬁguration and current location. The domains most contacted by such apps  
 were Alphabet, Facebook, Amazon, Microsoft and Adobe. Some pre-installed  
 apps, particularly in cheaper phones, have components in the system partition  
 that the user cannot easily remove, and which serve annoying ads or even act  
 as loaders for Trojans [1109].

**22.3.2** **Apple’s app ecosystem**

Apple has led from the start on security usability, providing ﬁne-grained access  
 controls long before Android, but its ecosystem has always been more closed.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 697 | Ross Anderson |

*22.3. PLATFORM SECURITY*

When the Mac was competing with the PC it was one hardware platform against  
 many OEMs; the same pattern followed with the iPod, where Apple demanded  
 30% of music sales, and it continued when Apple launched the iPhone. The  
 business model was much the same as a gaming console. Apple is the only

hardware vendor and demands 30% of software revenues, as well as 30% of in-  
 app purchases of online goods and services. Now that Apple has half the market  
 in developed countries (and three-quarters of teens) this is becoming a antitrust  
 issue. Every developer has horror stories, and although Amazon was allowed  
 in April 2020 to sell movies on Apple devices without giving Apple a cut [836],  
 this just highlights the arbitrary nature of Apple’s rules. Why should dating  
 sites like match.com have to hand it 30% of their sales, while Uber does not?  
 Apple treats dating as a digital good, but Uber tries to avoid taxi regulation  
 by claiming it’s the same, a mere matchmaking service between drivers and  
 riders. The rules appear to hit smaller ﬁrms particularly hard, and imposed an  
 ‘Apple tax’ on people like musicians, ﬁtness instructors and yoga teachers who  
 went online because of the pandemic, if people booked them via an iPhone app.  
 All this has led to an antitrust lawsuit in the USA from Epic Games, and a  
 competition policy investigation by the EU [888].

Apple also used its control of the hardware and the operating system to

implement rights-management mechanisms to protect its aftermarket revenue;  
 competing app stores are not allowed. The company does due diligence on

developers, requiring them to pay $99 a year for a license. Its app vetting

process is a lot tougher than Google’s: there’s extensive automated security  
 testing, followed by manual review to ensure that apps follow Apple policy  
 on matters such as payment, content and abuse. To support this, iOS apps  
 submitted to the App Store are only allowed to use the publicly-documented  
 APIs [1812]. Academic researchers have therefore dug into the iOS ecosystem a  
 lot less, but nevertheless a few things can be said.

The overall protection against malware is the best of any mass-market sys-

tem, with zero-day remote exploits of iOS trading for multiple millions of dollars  
 and being patched as soon as they’re used at scale. Indeed, when our own uni-  
 versity’s ﬁnance division has asked for advice on how to protect really high-value  
 transactions against phishing, my advice has been simple: buy an iPad on which  
 you run the bank’s authenticator app to release payments, use it only for pay-  
 ments, and keep it in a safe the rest of the time.

However, the protection isn’t entirely bulletproof, and various actors have

found workarounds.

First, there’s a long history of of hobbyists and others ‘jailbreaking’ Apple

devices, starting with people who objected to DRM or who wanted to sideload  
 their own apps without paying Apple $99 tax, as they can with Android. As  
 jailbreaks come out, Apple patches them; so at least the company has an in-  
 centive to patch its devices up to date, rather than abandon them after sale as  
 the typical Android OEM does. Sometimes patching isn’t possible, as when the  
 exploit is of the device’s boot ROM; for example, the 2019 Checkra1n jailbreak  
 will liberate most devices sold before 2017 [798], and the forensics industry uses  
 the Checkm8 jailbreak, which exploits the boot ROM of all iPhones from the  
 4S to the X [798]; this is used widely in the forensic ‘kiosks’ sold to the world’s  
 police forces, as I describe in section 26.5.1. Although ROM exploits cannot

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 698 | Ross Anderson |

*22.3. PLATFORM SECURITY*

defeat the user PIN on devices later than the 5s, thanks to the secure element,  
 they can access those user data that are made accessible after ﬁrst unlock, as  
 described in section 6.2.7. There’s also a market for carrier unlocking, where  
 you can also assume that the phone is in the physical custody of the attacker.

Attacks that can exploit iOS remotely are more valuable, as state actors are

willing to pay millions of dollars for them. We described in section 2.2.4 how  
 the UAE used such a tool to target dissidents, and how Saudi Arabia used one  
 against Jeff Bezos, whose newspaper the Washington Post they detested; the  
 Saudis also hacked their regional rival, the King of Qatar. Cybercriminals also  
 do it: in 2019, Google’s Project Zero revealed iOS exploits that were being used  
 in the wild to infect iPhones [204]. Apple always patches such exploits quickly,  
 so your millions only give you access to a handful of targets. If someone’s likely  
 to spend a million dollars to compromise your phone, you’d better have several  
 and not tell your enemies the number of your private phone that contains the  
 data you really care about9

Second, Apple sells large ﬁrms ‘enterprise certiﬁcates’ which let iOS develop-

ers bypass the app review process. This led to abuse and spats, with Facebook’s  
 enterprise cert being suspended until their app stopped infringing App store pol-  
 icy; Google’s app on the iPhone had a similar experience, and suddenly lots of  
 abuse by porn, gambling and spyware apps came to light. They had been abus-  
 ing enterprise certiﬁcates and hiding in plain sight in the app store [1697]. Many  
 of the bad actors had got their enterprise certs by pretending to be helpline apps  
 from MNOs in less developed countries [1170].

Third, Apple is like Android in that it doesn’t allow the user to block an

app’s access to the Internet. So we ﬁnd ﬁrewall apps for iOS too, but this is one  
 way in which the iOS privacy mechanisms get in the way of privacy. One app  
 can’t even see another let alone block it, so all the iOS ﬁrewalls can do on the  
 iPhone is block access to ad servers.

Although the malware issues are less serious than with Android, the same

market forces apply, and so ad abuse still happens. Many popular apps (in-  
 cluding dating apps such as Grindr and OkCupid) share a lot of data with  
 advertisers, and are still allowed in the Apple ecosystem [1762]. The same holds  
 for apps you might expect to be more privacy conscious, such as VPNs and ad  
 blockers – where the privacy exploits come in through embedded ad networks, as  
 in the Android ecosystem [1739]. In one case, an advertising SDK let its authors  
 steal clicks from the 1,200 apps that used it and were installed on 300m iPhones;  
 its code had stealth features that may have helped it past the app review pro-  
 cess [1314]. And although more apps are paid for in the Apple App Store than in  
 the Google Play Store (6% rather than 4.4%) and people assume that paid apps  
 that don’t show apps don’t track you, such an expectation may be optimistic –  
 in both ecosystems. In section 22.3.1.5 I mentioned research showing how the  
 paid versions of Android apps often still track you. One might expect similar  
 results for Apple, but the iPhone is a harder platform to do research on.

9I know of one tycoon who would borrow the mobile phone of a different employee each

day and get the switchboard to forward his calls. If that’s your strategy you’d better assume  
 it may occasionally double as a listening device and have your PA carry it for you. And

against a state adversary, maintaining separation between a hot phone and a cold one is not  
 straightforward: see the cotraveler system described in section 2.2.1.10.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 699 | Ross Anderson |

*22.3. PLATFORM SECURITY*

Apple, like Google, has been progressively tightening up the permissions

apps need. For example, iOS13 reﬁnes geodata from ‘allow’ on installation to  
 ‘allow once’ and ‘allow while using app’, and also curtails the use of wiﬁ and  
 Bluetooth to determine location – causing the same kind of complaints from  
 developers [434]. From September 2020, iOS14 will turn *identiﬁcation for ad-*  
 *vertisers* (IDFA) from opt-out to opt-in, essentially killing it, and undermining  
 advertisers’ ability to track the effectiveness of campaigns – supposedly for pri-  
 vacy, but it also looks set to promote Apple’s ad business at the expense of  
 Google, Facebook and third-party ad service ﬁrms [1073].

The two stores share some political problems, such as the fact that they both

allowed an app used by men in Saudi Arabia to control the movements of their  
 wives, daughters and servants, as I discussed in section 2.5.4. Occasionally, they  
 do diverge. Apple is more aggressive than Google at removing ‘bad’ apps, though  
 this can sometimes get them a bad press. During the 2019 protests in Hong  
 Kong, Apple banned a crowdsourced protest safety app that demonstrators were  
 using to avoid the police, claiming “Your app contains content – or facilitates,  
 enables, and encourages an activity – that is not legal ... speciﬁcally, the app  
 allowed users to evade law enforcement”, while Google left the Android version  
 up [1253].

Another political controversy arose with coronavirus contact tracing. In

|  |  |  |
| --- | --- | --- |
| February 2020 the government of Singapore announced an app that would use  Bluetooth to record which phones had been near each other, so that when  someone tested positive for the virus, public health o�cials could trace possible  contacts automatically rather than just asking the patient who they’d met over  the past week. This turned out to not work very well, as Bluetooth isn’t a good  ranging technology. If you set the volume to be sure to see people 2m away,  you see a fair number 10m away – which greatly increases the number of false  alarms that contact tracers have to deal with. What’s more, if the proportion  of the population running the app is *p* then the probability that both a patient  and their contact were both running it is *p*2 and the missed alarm rate is 1*�p*2;  was reported in April, a number of other countries, including the UK, France,  Germany, Latvia and Australia, had started to develop contact tracing apps  too. They discovered that the restrictions on Bluetooth use made such apps  tricky to write for Android phones and essentially impossible for iPhones [437].  When they asked for better access Google and Apple refused, citing the privacy  risk to their customers if all apps could do Bluetooth contact tracing. Google  and Apple made available an API for anonymous contact tracing, but from  the epidemiologists’ point of view this is even less useful [1799]. This led to  criticism of Google and especially Apple for taking policy decisions that are the  job of elected politicians [955]. Germany switched to the Google/Apple API  but started requiring pubs and restaurants to keep lists of customers’ contact  details, so that if one customer gets sick, people who sat nearby can be traced  using traditional methods. | | |
| **Security Engineering** | 700 | Ross Anderson |

*22.3. PLATFORM SECURITY*

**22.3.3** **Cross-cutting issues**

The convergence of the two ecosystems is leading to a growing number of cross-  
 cutting issues. These apply not just to phones but to other IoT devices, many  
 of which are either in the iOS ecosystem – such as Apple watches – or the  
 Android one – including thermostats, doorbell cameras, building sensors and  
 Google Home smart speakers. The other notable ecosystem is probably that of  
 the Amazon Alexa, which kickstarted the smart speaker product category (this  
 category has grown extremely quickly, taking 4 years to be adopted by half the  
 US population rather than 8 for the smartphone). Many of these devices are  
 also designed to support an ecosystem of apps, although the number and usage  
 varies by product.

In addition to the issues that stem from the MNOs, which we discussed in

section 22.2.5, and the rapacious ad ecosystems, which we discussed in the above  
 section, a major problem is poorly engineered apps.

Quite simply, when billions of people entrust their ﬁnancial lives, their social

lives and even their sex lives to apps, then poorly-written apps can cause real  
 harm. Speciﬁc application issues have been discussed in many other chapters  
 of this book. Here, one example may suffice to put things in context. It illus-  
 trates a problem that many app developers just don’t think through – that of  
 revocation. In fact, when assisting in the design of a payment app, we spent  
 about half of the security-engineering time working out in detail how we’d cope  
 with stolen phones: how payments could be blocked quickly when alerts came in  
 from different stakeholders, what would happen when the crime victim walked  
 into a shop the following day and bought a new phone, whether you’d rely  
 on the phone shop to authenticate them or make them call a bank contractor,  
 how you’d deal with phone OEMs who had their own backup and recovery ser-  
 vices – an absolute mass of mind-numbing detail. That’s what real engineering  
 comes down to: working with your supply chain and thinking through both the  
 customer experience and the possible abuse cases.

My example of what can happen when you don’t pay enough attention is

FordPass, an app that enables you to control a rental car so you can track it, lock  
 and unlock it, and start the engine – even several months after you’ve returned  
 it to the rental lot [794]. There are many more cases, but this is enough to  
 illustrate that poorly designed apps can expose other systems, including safety-  
 critical ones.

The threats from poorly written apps cover the whole spectrum of conﬁden-

tiality, integrity and availability. The consequences of goods relying on apps  
 that are no longer maintained are such that the EU passed the Sales of Goods  
 Directive in 2019 requiring vendors of goods with digital components to main-  
 tain these components for at least two years and for longer if that is a reasonable  
 expectation of the customer. From January 2022, phone apps supplied along  
 with a durable good such as a car or washing machine will have to be maintained  
 for ten years after the last of these products leaves the showroom. We’ll discuss  
 sustainability further in the last chapter of this book.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 701 | Ross Anderson |

*22.4. SUMMARY*

**22.4** **Summary**

Phone security is a fascinating case study. People have been cheating phone  
 companies for a century, and since deregulation the phone companies have been  
 vigorously returning the compliment. To start off with, systems were not really  
 protected at all, and it was easy to evade charges and redirect calls. The mech-  
 anism adopted to prevent this – out-of-band signalling – proved inadequate as  
 the rapidly growing complexity of the system opened up many more vulnerabil-  
 ities. These range from social engineering attacks on users through poor design  
 and management of terminal equipment such as PBXes to the exploitation of  
 various hard-to-predict feature interactions. The main disruptive force was the  
 development of premium-rate services that enabled people to steal real money.

On the mobile front, the attempts to secure GSM and its third, fourth and

ﬁfth generation successors make an interesting case study. Their engineers con-  
 centrated on communications security threats rather than computer security  
 threats, and on the phone companies’ interests at the expense of the customers’.  
 Their efforts were not entirely in vain but have led to an immensely complex  
 global ecosystem that has become the subject of signiﬁcant political tussles,  
 particularly over the control of 5G infrastructure.

The dominating factor in 2020 is the mobile app ecosystems. The Android

ecosystem has attracted hundreds of thousands of developers, ranging from ﬁrms  
 like Uber that have built apps into major international businesses, through apps  
 offered by many established businesses and a host of specialist tools, to a sub-  
 stantial criminal fringe. The Apple ecosystem is more regulated but similar in a  
 number of respects. Many apparently innocuous apps in both ecosystems can be  
 abused in interesting ways, and the ad networks they use are a pervasive threat  
 to privacy. The ecosystems of mobile apps, apps on more traditional platforms  
 such as laptops, and apps on devices such as watches and cars converge and  
 overlap in various ways, but insofar as they are still distinct, mobile platforms  
 protect apps from each other more robustly than laptops do and the platform  
 operators make signiﬁcant security efforts at the ecosystem level. Indeed, as  
 most Android phones are not patched up to date and are therefore insecure,  
 the heavy lifting isn’t done at the level of technical platform security but at the  
 level of the ecosystem.

**Research Problems**

The interaction between communications, mobility, platforms, and apps contin-  
 ues to be fertile ground for both interesting research and expensive engineering  
 errors. We have explored a lot of the issues over the past ten years in the mobile  
 phone app ecosystem, mostly in the Android part of it where most of the prob-  
 lems occur. Mobility is now extending to all sorts of other devices, from your  
 watch to your car, and many of the issues around app ecosystems are arising  
 with smart speakers and other domestic devices. Given the sheer scale of these  
 new emerging ecosystems, we will need innovative ways to automate the hunt  
 for both threats and vulnerabilities. One approach is to build honeypots and  
 look for attack traffic; a somewhat more forward defence may be to analyse

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 702 | Ross Anderson |

*22.4. SUMMARY*

the companion apps used to control IoT devices and infer vulnerabilities from  
 them [1978].

**Further Reading**

Information about the world’s phone systems is scattered across a large number  
 of standards documents that can be rather heavy going, while app platforms at  
 least have official guides, white papers and developer communities. Keeping up  
 with the latest exploits is a matter of following the security blogs and tech press.  
 There are some good surveys of speciﬁc subproblems, which I’ve cited in the  
 relevant sections, but I’m not aware of any good books or survey papers of the  
 overall phone security scene. Perhaps that’s inevitable; now that more people go  
 online via mobile devices then from laptops or desktops, mobile security touches  
 one way or another on much of the subject matter of this book.

|  |  |  |
| --- | --- | --- |
| **Security Engineering** | 703 | Ross Anderson |