CHAPTER

18

Tamper Resistance

It is relatively easy to build an encryption system that is secure if it is working as intended and is used correctly but it is still very hard to build a system that does not compromise its security in situations in which it is either misused or one or more of its sub-components fails (or is 'encouraged' to misbehave) … this is now the only area where the closed world is still a long way ahead of the open world and the many failures we see in commercial cryptographic systems provide some evidence for this.

– BRIAN GLADMAN

The amount of careful, critical security thinking that has gone into a given security device, system or program is inversely proportional to the amount of high-technology it uses.

– ROGER JOHNSTON

18.1 Introduction

Tamper-resistant devices are everywhere now. Examples we've discussed so far include:

- the EMV chips used in bank cards and the SIMs used in mobile phones for *authentication*;
- the contactless cards used as transport tickets and the smartcards used in pay-TV decoders for *service control*;
- chips used for *accessory control* in printer toner cartridges and game-console accessories;
- the TPM chips in phones, laptops and servers to provide a *root of trust* to support secure boot and hard-disk encryption;
- hardware security modules used to encrypt bank PINs, not just in bank server farms but in ATMs and some point-of-sale terminals;
- the NFC chips used in Android phones to store contactless payment credentials, and the enclave chips in iPhones that store your fingerprint and crypto keys;

- cryptographic modules buried in vending machines that sell everything from railway tickets through postage stamps to the magic numbers that activate your electricity meter;
- various chips used for *manufacturing control* by firms who want to have their products made by low-cost overseas manufacturers but don't want to see extra products made without their consent on a 'third shift' and sold on the grey market.

Many of the devices on the market are insecure. In section 4.3.1 I described how reverse engineering remote key entry devices for cars led to class breaks that notably increased car theft; in section 13.2.5 I described how reverse engineering the Mifare card compromised many building locks and transport ticketing systems; and in section 12.6.1.1, I described card payment terminals that could be compromised trivially, leading to card counterfeiting and transaction manipulation attacks.

Yet some are pretty good. The best cryptographic modules used in banking and government withstand all known types of physical attack, and can only be defeated when people either run insecure software on them or rely on insecure devices to interface with users. Smartcard tamper resistance has evolved in a long war between pay-TV pirates cloning subscriber cards and the pay-TV industry trying to stop them, and was honed in an arms race between firms that wanted to lock down their products, and others who wanted to unlock them. The tussles over printer cartridges were important here, as both the printer makers who were trying to control aftermarkets, and the independent cartridge makers who were trying to break into these markets, are acting lawfully. Other hackers work for lawyers, reverse engineering products to prove patent infringements. There are academics who hack systems for glory, and to push forward the state of the art. And finally there are lots of grey areas. If you find a way to unlock a mobile phone, so that it can be used on any network, is that a crime? It depends on how you do it, and on what country you're in.

Given the wide range of products and the huge variation in quality, the security engineer needs to understand what tamper resistance is, and what it can and can't do. In this chapter I'm going to take you through the past thirty years of evolution of attack and defence.

If a computer cannot resist physical tampering, an attacker can simply change the software. Computers in data centres are protected by physical barriers, sensors and alarms. And an ATM is basically a PC in a safe with banknote dispensers and alarm sensors, often bolted to a wall or a plinth.

Where tamper resistance is needed purely for integrity and availability, it can sometimes be implemented using replication on different servers that perform transactions simultaneously and vote on the result; this is being reinvented nowadays with blockchains and other consensus protocols. The threshold schemes discussed in section 15.4 can also provide confidentiality

for key material. But tamper-resistant devices can provide confidentiality for the data too, and the arrival of CPUs that support enclaves such as SGX and TrustZone hold out the prospect of computing with encrypted data in cloud services.

18.2 History

The use of tamper resistance in cryptography goes back centuries [1003]. Naval codebooks were weighted so they could be thrown overboard if capture was imminent; the dispatch boxes used by British government ministers' aides to carry state papers were lead-lined to make sure they'd sink. Codes have been printed in water-soluble ink; Russian one-time pads were printed on cellulose nitrate, so they'd burn furiously if lit; and one US wartime cipher machine came with self-destruct thermite charges. But key material was often captured in surprise attacks, so attempts were made to automate the tamper response process. Some mechanical cipher machines were built so that opening the case erased the key settings, and early electronic devices followed suit.

After the notorious Walker family sold US Navy key material to the Russians for over 20 years [878], engineers paid more attention to the question of how to protect keys in transit too. The goal was 'to reduce the street value of key material to zero', and this can be achieved either by *tamper resistant* devices from which the key cannot be readily extracted, or *tamper evident* ones from which key extraction would be obvious.

Paper keys were once carried in 'tattle-tale containers', designed to show evidence of tampering. When electronic key distribution came along, a typical solution was the 'fill gun': a portable device that dispenses crypto keys in a controlled way. Nowadays the physical transport of crypto key material usually involves a smartcard, or a similar chip packaged as a key. Your SIM card and bank card are just the most visible examples. The control of key material also acquired broader purposes, with both the US and the UK governments using it to restrict their networks to approved devices. Live key material would only be supplied once the system had been properly accredited.

Once initial keys have been loaded, further keys may be distributed using authentication protocols. Our subject here is the physical defenses against tampering.

18.3 Hardware security modules

The IBM 4758 (Figures 18.1 and 18.2) was the leading commercial cryptographic processor in the early 2000s, and is important for four reasons. First,

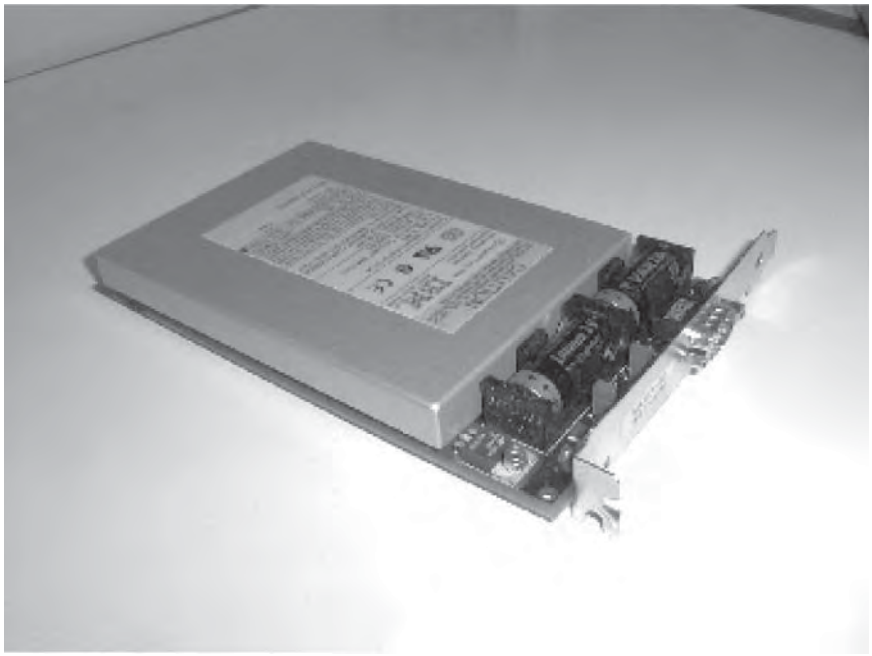

Figure 18.1: The IBM 4758 cryptoprocessor (courtesy of Steve Weingart)

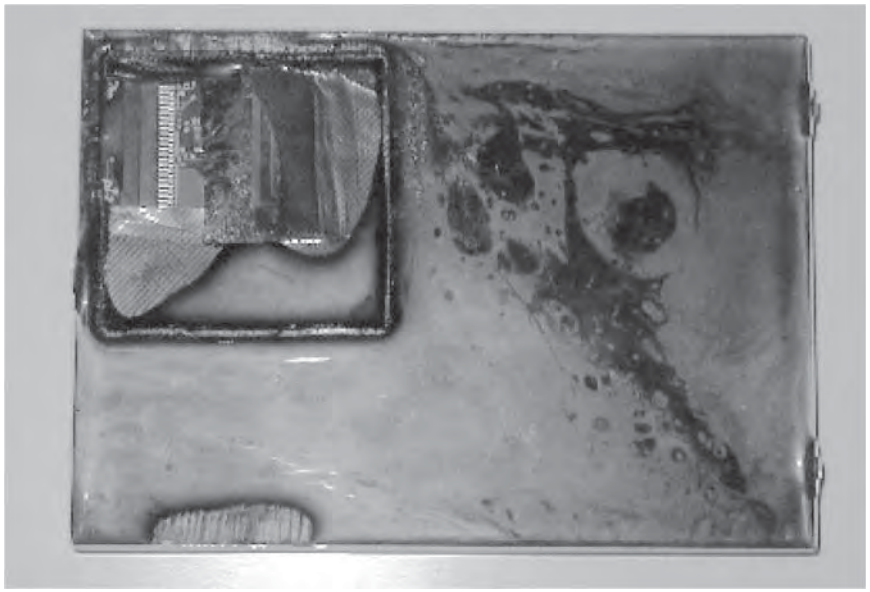

Figure 18.2: The 4758 partially opened showing (from top left downward) the circuitry, aluminium electromagnetic shielding, tamper-sensing mesh and potting material (courtesy of Frank Stajano)

it was the first commercial product to be evaluated to the highest level of tamper resistance (FIPS 140-1 level 4) [1401] then set by the US government. Second, there is an extensive literature about it, including its history, hardware and software [1646, 2002, 2005]. Third, it was therefore a high-profile target, and from 2000–2005 my students and I put a lot of effort into attacking it and understanding the residual vulnerabilities. Fourth, the current IBM flagship product, the 4765, isn't hugely changed except for fixing some of the bugs we found.

The back story starts in the 1970s, when Mikhail Atalla had the idea of a black-box cryptographic module to manage bank PINs. As early cryptographic schemes for ATMs were rather weak, IBM developed a better block cipher that became the Data Encryption Standard, as described in Chapter 5. There followed a period of intense research about precisely how block ciphers could be used to manage PINs in a single bank, and then in a network of many banks [1303]. The banking community realised that commercial operating systems were likely to remain insufficient to protect PINs, particularly from bank insiders, and decided to use separate hardware to manage them.

This led to the development of standalone cryptographic modules or *hardware security modules* (HSMs), as fintech people call them. These are microcomputers encased in robust metal enclosures, with encryption hardware and special *key memory*, static RAM that is zeroized when the enclosure is opened. Initially, this just involved wiring the power supply to the key memory through a number of lid switches. So whenever the maintenance crew came to replace batteries, they'd open the lid and destroy the keys. Once they'd finished, the HSM custodians would reload the key material. In this way, the HSM's owner could hope that its keys were under the unique control of its own trustworthy staff.

How to hack a cryptoprocessor (1)

The obvious attack is just to steal the keys. In early HSMs, the master keys were kept in PROMs that were loaded into a special socket in the device to be read during initialization, or as strings of numbers that were typed in at a console. The PROMs could be pocketed, taken home and read out. Cleartext paper keys were even easier: just scribble down a copy.

The fix was shared control – to have two or three master key components, and make the actual master key by combining them. The PROMs (or paper keys) would be kept in different safes under the control of different departments. This taught us that shared control is a serious security usability hazard. The manual may tell the custodians to erase the live keys, let the engineer fix the device, and then re-load the keys afterwards. But many senior men used to think that touching keyboards was women's work, and even today they think that technical work is beneath them. And who reads the manual anyway? So managers often give both keys to the engineer to save the bother. In one case,

a dishonest engineer got them to enter the keys using a laptop that acted as a terminal but had logging switched on [55]. I've even come across cases of paper master keys for an automatic teller machine being kept in the correspondence file in a bank branch, where any of the staff could look them up.

How to hack a cryptoprocessor (2)

Early devices were vulnerable to attackers cutting through the casing. Second-generation devices made physical attacks harder by adding photocells and tilt switches. But the difficult opponent is the maintenance engineer – who could disable the sensors on one visit and extract the keys on the next.

By about 2000, the better products separated all the components that can be serviced (such as batteries) from the core of the device (such as the tamper sensors, cryptoprocessor, key memory and alarm circuits). The core was then potted into a solid block of a hard, opaque substance such as epoxy. The idea was that any physical attack would involve cutting or drilling, which could be detected by the guard who accompanies the engineer into the bank computer room[1]. At least it should leave evidence of tampering after the fact. This is the level of protection needed for medium-level evaluations under the FIPS standard.

How to hack a cryptoprocessor (3)

However, if a competent attacker can get unsupervised access to the device for even a short period of time – and, to be realistic, that's what the maintenance engineer probably has, as the guard doesn't understand what's going on – then potting the device core is inadequate. For example, you might scrape away the potting with a knife and drop the probe from a logic analyzer on to one of the chips. In theory, scraping the sticky epoxy should damage the components inside; in practice, it's just a matter of patience. Cryptographic algorithms such as RSA, DES and AES have the property that an attacker who can monitor any bitplane during the computation can recover the key [861].

So the high-end products acquired a tamper-sensing barrier. An early example appeared in IBM's μABYSS system in the mid-1980s, which used loops of 40-gauge nichrome wire wound loosely around the device as it was embedded in epoxy, and then connected to a sensing circuit [2002]. The theory was that techniques such as milling, etching and laser ablation would break the wire, erasing the keys. But the wire-in-epoxy technique can be vulnerable to slow erosion using sand blasting; when the sensing wires become visible at the surface of the potting, shunts can be connected round them. In 2018 Sergei Skorobogatov managed to use a combination of acid etching and masking to expose a battery-powered chip, on the Vasco Digipass 270, showing that given

[1]That at least was the theory; experience suggests it's a bit much to ask a minimum-wage guard to ensure that a specialist in some exotic piece of equipment repairs it using some tools but not others.

decent lab technique you can indeed attack live circuits protected by wires in epoxy [1785].

The next major product from IBM, the 4753, used a metal shield combined with a membrane printed with a pattern of conductive ink and surrounded by a more durable material of similar chemistry. The idea was that any attack would break the membrane with high probability. The 4758 had an improved tamper-sensing membrane in which four overlapping zig-zag conducting patterns were doped into a urethane sheet, which was potted in a chemically similar substance so that an attacker cutting into the device had difficulty even detecting the conductive path, let alone connecting to it. This potting surrounds the metal shielding, which in turn contains the cryptographic core. The design is described in more detail in [1799].

How to hack a cryptoprocessor (4)

The next class of attack exploited *memory remanence*, the fact that many kinds of computer memory retain some trace of data that have been stored there. Once a certain security module had run for some years using the same master keys, their values *burned in* to the device's static RAM. On power-up, about 90% of the relevant memory bits would assume the values of the previously stored secret keybits, which was quite enough to recover the keys [108]. Memory remanence affects not just static and dynamic RAM, but other storage media too. The relevant engineering and physics issues are discussed in [838] and [841], and in 2005 Sergei Skorobogatov discovered how to extract data from Flash memory in microcontrollers, even after it had been 'erased' several times [1774]; like it or not, the wear-levelling processors in Flash chips become part of your trusted computing base. RAM contents can also be *burned in* by ionising radiation, so radiation sensing or hardening might make sense too.

How to hack a cryptoprocessor (5)

Computer memory can also be frozen by low temperatures. By the 1980s it was realized that below about −20° C, static RAM contents can persist for several seconds after power is removed. This extends to minutes at the temperatures of liquid nitrogen. So an attacker might freeze a device, remove the power, cut through the tamper sensing barrier, extract the RAM chips containing the keys, and power them up again in a test rig.

In 2008, Alex Halderman and colleagues developed this into the *cold boot attack* on encryption keys in PCs and phones [855]. Modern DRAM retains memory contents for several seconds after power is removed, and even longer at low temperatures; by chilling memory with a freezing spray, then rebooting the device with a lightweight operating system, keys can often be read out. Software encryption of disk contents can be defeated unless there are mechanisms to zeroise the keys on power-down. Even keeping keys in special hardware such as a TPM isn't enough if all it's doing is limiting the number of

times you can guess the hard disk encryption password, but then copying the master key to main memory once you get the password right so that the CPU can do the rest of the work. You need to really understand what guarantees the crypto chip is giving you – a matter we'll discuss at greater length in the chapter on advanced cryptographic engineering.

Anyway, the better cryptographic devices have temperature and radiation alarms. But modern RAM chips exhibit a wide variety of memory remanence behaviors; remanence seems to have got longer as feature sizes have shrunk, and in unpredictable ways even within standard product lines. So although your product might pass a remanence test using a given make of SRAM chip, it might fail the same test with the same make of chip purchased a year later [1772]. This shows the dangers of relying on a property of some component to whose manufacturer this property is unimportant.

The main constraints on the HSM alarms are similar to those we encountered with more general alarms. There's a tradeoff between the false alarm rate and the missed alarm rate, and thus between security and robustness. Vibration, power transients and electromagnetic interference can be a problem, but temperature is the worst. A device that self-destructs if frozen can't be sent reliably through normal distribution channels, as aircraft holds can get as low as −40°C. (We've bought crypto modules on eBay and found them dead on arrival.) Military equipment makers have the converse problem: their kit must be rated from −55° to +155° C. Some military devices use protective detonation; memory chips are potted in steel cans with a thermite charge precisely calculated to destroy the chip without causing gas release from the can. Meeting simultaneous targets for tamper resistance, temperature tolerance, radiation hardening, shipping safety, weight and cost can be nontrivial.

How to hack a cryptoprocessor (6)

The next set of attacks on cryptographic hardware involves monitoring the RF and other electromagnetic signals emitted by the device, or even injecting signals into it and measuring their externally visible effects. This technique, which is variously known as 'Tempest', 'power analysis,' 'side-channel attacks' or 'emission security', is such a large subject that I devote the next chapter to it.

As far as the 4758 was concerned, the strategy was to have solid aluminium shielding and to low-pass filter the power supply to block the egress of any signals at the frequencies used internally for computation. This shielding is inside the tamper-sensing membrane, to prevent an opponent cutting a slot that could function as an antenna.

How to hack a cryptoprocessor (7)

We never figured out how to attack the hardware of the 4758. The attacks we have seen on high-end systems have involved the exploitation of logical rather than physical flaws. One hardware security module, the Chrysalis-ITS

Luna CA3, had its key token's software reverse engineered by Mike Bond, Daniel Cvrček and Steven Murdoch who found code that enabled an unauthenticated "Customer Verification Key" to be introduced and used to certify the export of live keys [284]. Most recently, in 2019, Gabriel Campana and Jean-Baptiste Bédrune found a buffer overflow attack on the Gemalto Safenet Protect Server PSI-E2/PSE2 by fuzzing the HSM emulator that came with its software development kit, then checked this on a real HSM, and wrote code to upload arbitrary firmware, which is persistent and can download all the secrets [204].

This did not happen to IBM's 4758, which had a formally verified operating system. But most of its users ran a banking crypto application called CCA that is described in [917]. Mike Bond and I discovered that the application programming interface (API) that CCA exposed to the host contained a number of exploitable flaws. The effect was that a programmer with access to the host could send the security module a series of commands that would cause it to leak PINs or keys. These vulnerabilities were largely the legacy of previous encryption devices with which 4758 users needed to be backward compatible, and in fact most other security modules were worse. Such attacks were hard to stop, as from time to time Visa would mandate new cryptographic operations to support new payment network features and these would introduce new systemic vulnerabilities across the whole fleet of security modules [22]. Some HSMs now have two APIs: an internal one which the vendor tries to keep clean (but which needs to have the ability to import and export keys) and an external one that implements the standards of whatever industry the HSM is being used to support. The software between the two APIs may be trusted, but can be hard to make trustworthy if the external API is insecure. In effect, it has to anticipate and block API attacks. The end result is that many banks pay top dollar for secure HSMs which they use for formal compliance, while relying on other access control mechanisms to shield these precious devices from attack. There are even specialist firms selling firewalls to shield HSMs from software-based harm. I'll discuss API attacks in detail in the chapter on advanced cryptographic engineering.

18.4 Evaluation

A few comments about the evaluation of HSMs are in order before we go on to discuss cheaper devices. When IBM launched the 4753 in 1991, they proposed the following classification of attackers in the associated white paper [9]:

1. Class 1 attackers – 'clever outsiders' – are often very intelligent but may have insufficient knowledge of the system. They may have access to only moderately sophisticated equipment. They often try to take advantage of an existing weakness in the system, rather than try to create one.

2. Class 2 attackers – 'knowledgeable insiders' – have substantial specialized technical education and experience. They have varying degrees of understanding of parts of the system but potential access to most of it. They often have highly sophisticated tools and instruments for analysis.
3. Class 3 attackers – 'funded organizations' – are able to assemble teams of specialists with related and complementary skills backed by great funding resources. They are capable of in-depth analysis of the system, designing sophisticated attacks, and using the most advanced analysis tools. They may use Class 2 adversaries as part of the attack team.

Within this scheme, the typical microcontroller is aimed at blocking clever outsiders; the early 4753 aimed at stopping knowledgeable insiders, and the 4758 was aimed at (and certified for) blocking funded organizations. This classification is now a bit dated; we see class 1 attackers renting access to class 3 equipment, while class 3 attackers nowadays are not just national labs, but commercial competitors and even university security teams. In our case, we have people with backgrounds in maths, physics, software and banking, and we've had friendly manufacturers giving us samples of their competitors' products for us to break.

The FIPS certification scheme is operated by laboratories licensed by the US government. The original 1994 standard, FIPS 140-1, set out four levels of protection, with level 4 being the highest, and this remained in the next version, FIPS 140-2, which was introduced in 2001. There was a huge gap between level 4 and level 3; devices at that level were often easy for experts to attack. In fact, the original paper on evaluation by IBM engineers proposed six levels [2005]; the FIPS standard adopted the first three of these as its levels 1–3, and the proposed level 6 as its level 4 (the 4758 designer Steve Weingart tells the story in [2004]). The gap, commonly referred to as level 3.5 or 3+, is where many of the better commercial systems were aimed from the 1990s through 2019. Such equipment attempts to keep out the class 1 attack community, while making life hard for class 2 and expensive for class 3.

There was about a decade of consultation about whether to abandon FIPS 140 in favour of ISO 19790 – a move supported by vendors, particularly those outside the USA. Critics of the FIPS approach noted that it didn't cover non-invasive security such as buffer overflows and API attacks; that its concept of roles was tied to human actors in companies, rather than other system components; that it failed to cover some methods of side-channel analysis; that it was generally aimed at outdated technology; that the FIPS standard includes the dual elliptic curve deterministic random bit generator, known to contain an NSA backdoor; and that it was changed too often by NIST issuing implementation guidelines, rather than by updating the standard regularly [1412]. Eventually, the US Department of Commerce gave

up and approved an updated version, FIPS 140-3, which simply refers to the ISO standards 19790:2012 and 24759:2017, and specifies some refinements. This came into force in September 2019 and in 2021 testing under FIPS 140-2 will cease.

18.5 Smartcards and other security chips

While there are tens of thousands of HSMs in use, there are billions of self-contained one-chip crypto modules containing nonvolatile memory, I/O, usually a CPU, often some specialised logic, and mechanisms to protect memory from being read out. Most are packaged as cards, while some look like physical keys. They range from transport tickets at the low end, through smartcards and the TPMs that now ship with most computers and phones, up to pay-TV cards and accessory control chips designed to withstand attack by capable motivated opponents for as long as possible.

Many attacks have been developed; we discussed the consequences of the breaks of the Mifare cards in section 13.2.5 and of car keys in section 4.3.1. Pay-TV subscriber cards in particular have been subjected to intensive attacks as they often have a universal shared secret key, so a compromise enables an attacker to make lots of counterfeit cards, while a break of a bank smartcard only lets the attacker loot that specific bank account. The accessory control chips in printer cartridges also protect a lot of 'value', and have driven real innovation in both attack and defence. I'll describe both pay-TV and accessory control in the chapter on copyright; in this section, I'll tell the story of how chip-level security evolved.

18.5.1 History

Smartcards were developed in France from the mid-70s to mid-80s; for the early history, see [833]. From the late 1980s, they started to be used at scale, initially as the *subscriber identity modules* (SIMs) in GSM mobile phones and as subscriber cards for satellite-TV stations. They started being used as bank cards in France and South Africa in 1994, followed by trials in the UK and Norway; this led to the EMV standard I mentioned in the chapter on banking and bookkeeping, with deployment in the rest of Europe from 2003 and the USA from about 2015.

A smartcard is a self-contained microcontroller, with a microprocessor, memory and a serial interface integrated in a single chip and packaged in a plastic card. Smartcards used in banking use a standard-size bank card, while in modern mobile phones a much smaller size is used. Smartcard chips are also packaged in other ways. In the STU-III secure telephones used in

the US government from 1987–2009, each user had a 'crypto ignition key', packaged to look and feel like a physical key; some prepayment electricity meters and pay-TV set-top boxes used the same approach. The TPM chips built into computer motherboards to support trusted boot are basically smartcard chips with an added parallel port, so the TPM can verify that the right software is being used to start up the computer. Contactless smartcards contain a smartcard chip plus a wire-loop antenna; most car keys are a slightly more complex version of the same idea, with an added battery to give greater range. In what follows I'll mostly disregard the packaging form factor and just refer to single-chip cryptographic modules as 'smartcards' or 'chipcards'.

Apart from bank cards, the single most widespread application is the mobile phone SIM. The handsets are personalized for each user by the SIM, which contains the key with which you authenticate yourself to the network. The strategy of using a cheap card to personalise a more expensive electronic device is found in other applications from pay-TV set-top boxes to smart meters. The device can be manufactured in bulk for global markets, while each subscriber gets a card to pay for service. The cards can also be replaced relatively quickly and cheaply in the event of a successful attack.

18.5.2 Architecture

The typical smartcard consists of a single die of up to 25 square millimeters of silicon containing a microprocessor (larger dies are more likely to break as the card is flexed). Cheap products have an 8-bit processor such as an 8051 or 6805, and the more expensive products have either a modular multiplication circuit to do public-key cryptography, or a 32-bit processor such as an Arm, or indeed both (hardware crypto is easier to protect against side-channel attacks). The high-end ones also tend to have a hardware random number generator. There's also serial I/O and a hierarchy of memory – ROM or Flash to hold the program and immutable data, Flash or EEPROM to hold customer data such as the user's account number, crypto keys, PIN retry counters and value counters; and RAM to hold transient data during computation.

The memory is limited by the standards of normal computers; outside the device, the only connections are for power, reset, a clock and a serial port. The physical, electrical and low-level logical connections, together with a file-system-like access protocol, are specified in ISO 7816. There are several main software architectures on offer, including at the bottom end the *Application Programming Data Units* (APDUs) defined by ISO 7816 which allow a reader to invoke specific applications directly, through the Multos operating system, to JavaCard which lets the card run apps written in a subset of the Java language, and which you (and your opponents in the underground) can

use to code up custom apps[2]. You can even buy overlay SIMs – smartcards 160 microns thick with contacts top and bottom, which you can program in JavaCard to carry out middleperson attacks on other smartcards (you stick the overlay on top of the target device).

18.5.3 Security evolution

When I first heard a sales pitch from a smartcard vendor – in 1986 when I was working as a banker – I asked how come the device was secure. I was assured that since the machinery needed to make the card cost $20m, just as for making banknotes, the system must be secure. I didn't believe this but didn't have the time or the tools to prove the salesman wrong. I later learned from industry executives that none of their customers were prepared to pay for serious security until about 1995; until then they relied on the small size of the devices, the obscurity of their design, and the inaccessibility of chip testing tools to make attacks more difficult. In any case, so long as they were only used for SIM cards, there were no capable motivated opponents. All I can achieve by hacking my SIM card is the ability to charge calls to my own account.

The application that changed this was satellite TV. TV operators broadcast their signals over a large footprint – such as all of Europe – and give each subscriber a card to compute the keys needed to decipher the channels they've paid for. Since the operators had usually only bought the rights to the movies for one or two countries, they couldn't sell subscriber cards elsewhere. This created a black market, into which forged cards could be sold. A critical factor was that 'Star Trek', which people in Europe had picked up from UK satellite broadcasts for years, was suddenly encrypted in 1993. In some countries, such as Germany, it wasn't available legally at any price. This motivated a lot of keen young computer science and engineering students to look for vulnerabilities. A further factor was that some countries, notably Ireland and Canada, didn't have laws yet against selling forged pay-TV cards; Canada didn't do this until 2002. So hackers could sell their wares openly.

This rapidly had knock-on effects. The first large financial fraud reported to involve a cloned smartcard was about a year later, in February/March 1995. The perpetrator targeted a card used to give Portuguese farmers rebates on fuel, conspiring with petrol stations who registered other fuel sales to the bogus cards in return for a share of the proceeds. The proceeds were reported to have been about $30m [1332].

How to hack a smartcard (1)

The earliest hacks targeted the protocols rather than the cards themselves. For example, some early pay-TV systems gave each customer a card with access

[2]JavaCard has quietly become one of the most widely deployed operating systems in the world with over 6 billion cards sold [1252].

to all channels, and then sent messages over the air to cancel those channels to which the customer hadn't subscribed after an introductory period. This opened an attack in which a device was inserted between the smartcard and the decoder to intercept and discard any messages addressed to the card. So you could cancel your subscription without the vendor being able to cancel your service. The same kind of attack was launched on the German phone card system, with handmade chip cards sold in brothels and in hostels for asylum seekers [185,1817].

How to hack a smartcard (2)

As smartcards use an external power supply, and store security state such as crypto keys and value counters in EEPROM, an attacker could freeze the EEPROM contents by removing the programming voltage, V_{PP}. Early smartcards received V_{PP} from the card reader on a dedicated contact. So by covering this contact with sticky tape, cardholders could prevent a value counter from being decremented. With some payphone chipcards, this gave infinite units.

The fix was to generate V_{PP} internally from the supply voltage V_{CC} using a voltage multiplier. However, this isn't foolproof as the circuit can be destroyed by an attacker, for example with a laser shot. As well as bypassing value controls, they can also bypass a PIN retry counter and try every possible PIN, one after another. So a prudent programmer won't just ask for a customer PIN and decrement the counter if it fails. You decrement the counter, check it, get the PIN, verify it, and if it's correct then increment the counter again[3].

How to hack a smartcard (3)

Another early attack was to read the voltages on the chip surface using a scanning electron microscope (SEM). The low-cost SEMs found in universities back then couldn't do voltage contrast microscopy at more than a few tens of kilohertz, so attackers would slow down the clock. In one card, attackers found they read out RAM contents with a suitable transaction after reset, as working memory wasn't zeroized.

Modern smartcard processors have a watchdog timer or other circuit to detect low clock frequency and reset the card, or else use dynamic logic. And the attacker could sometimes single-step the program by repeatedly resetting the card and clocking it n times, then n+1 times, and so on. But as with burglar alarms, there's a tradeoff between false alarms and missed alarms. Cheap card readers can have wild fluctuations in clock frequency when a card is powered up, causing many false alarms. Eventually, cards acquired an internal clock.

[3]Such *defensive programming* was common in the early days of computing, when computers used valves rather than transistors and used to break down every few hours. Back then, if you masked off three bits, you'd check the result was no more than seven, just to make sure.

How to hack a smartcard (4)

Once pay-TV operators had blocked the easy attacks, pirates turned to physical probing. Early smartcards had no protection against physical tampering except the microscopic scale of the circuit, a thin glass *passivation layer* on the surface of the chip, and potting that is typically some kind of epoxy. Techniques for depackaging chips are well known, and discussed in detail in standard works on semiconductor testing, such as [198]. In most cases, a milliliter of fuming nitric acid is more than enough to dissolve the epoxy.

Probing stations consist of microscopes with micromanipulators attached for landing fine probes on the surface of the chip. They are used in the semiconductor industry for testing production-line samples, and can be bought second-hand (see Figure 18.3). They may have specialized accessories, such as a laser to shoot holes in the chip's passivation layer.

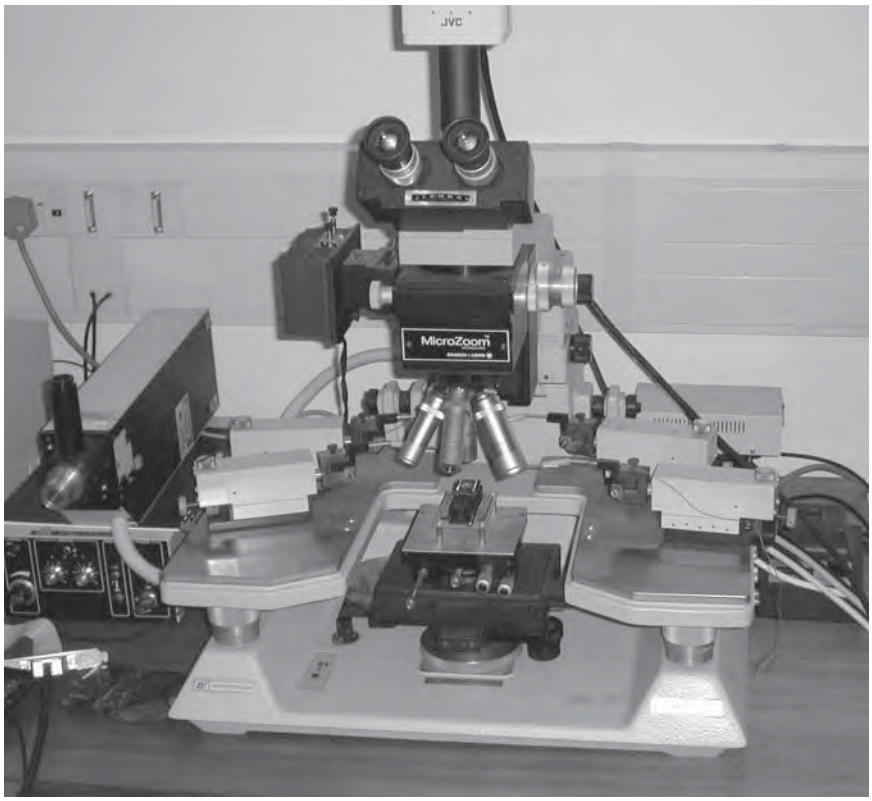

Figure 18.3: Our probing station

The usual target of a probing attack is the processor's bus. If the bus traffic can be recorded, this gives a trace of the program's operation. (It was once a recommended industry practice for the card to compute a checksum on memory

immediately after reset – giving a complete listing of all code and data.) So the attacker will find the bus and expose it for probing (see Figure 18.4). If the chip is using algorithms like AES and RSA, then unless there's some defense mechanism that masks the computation, a trace from even a single bus line will be enough to reconstruct the key [861].

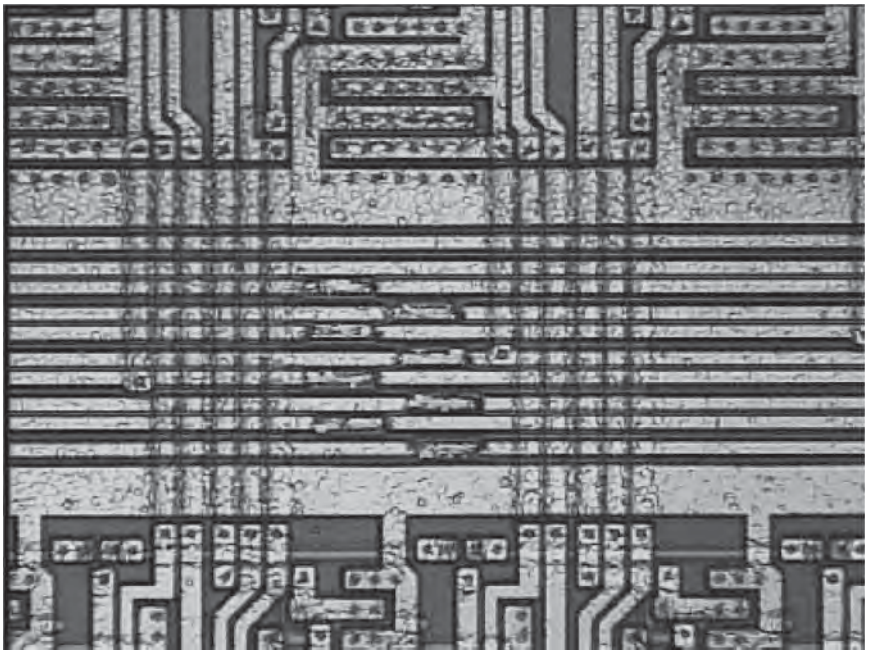

Figure 18.4: The data bus of an ST16 smartcard prepared for probing by excavating eight trenches through the passivation layer with laser shots (courtesy Oliver Kömmerling)

The first defense used by the pay-TV card industry was to endow each card with multiple keys or algorithms, and arrange things so that only those in current use would appear on the processor bus. Whenever pirate cards appeared on the market, a command would be issued over the air to cause legitimate cards to activate new keys or algorithms from previously unused memory. In this way, the pirates' customers would suffer a loss of service until the attack could be repeated and new pirate cards or updates could be distributed [2067].

How to hack a smartcard (5)

This strategy was defeated by Oliver Kömmerling's *memory linearization attack* in which the analyst damages the chip's instruction decoder in such a way that instructions such as jumps and calls – which change the program address other than by incrementing it – are broken [1080]. One way to do this is to drop a grounded microprobe needle on the control line to the instruction latch, so that whatever instruction happens to be there on power-up is executed repeatedly. The memory contents can now be read off the bus. In fact, once some

of the device's ROM and EEPROM is understood, the attacker can skip over unwanted instructions and cause the device to execute only instructions of their choice. So with a single probing needle, they can get the card to execute arbitrary code, and in theory could get it to output its secret key material on the serial port. This can be thought of as an early version of the return-oriented programming attack. But probing the memory contents off the bus is usually more convenient.

There are often several places in the instruction decoder where a grounded needle will prevent programmed changes in the control flow. So even if it isn't fully understood, memory linearization can often be achieved by trial and error. One particularly vulnerable smartcard family was the Hitachi H8/300 architecture, which had a 16-bit bus with the property that if the most significant bit equals 1 then the CPU will always execute single-cycle instructions without any branches. So by shooting the MSB bus line with a laser, the memory could be easily read out [1785]. Other CPUs based on RISC cores also tend to suffer from this. Some of the more modern processors have traps which prevent memory linearization, such as watchdog timers that reset the card unless they themselves are reset every few thousand instructions.

Memory linearization is an example of a *fault induction attack*. There are many other examples. Faults can be injected into processors in many ways, from hardware probing through power transients and laser illumination. One common target is the test circuitry. A typical chip has a self-test routine in ROM that is executed in the factory and allows all the memory contents to be read and verified. In some cases, a fuse is blown in the chip to stop an attacker using the facility. But the attacker can cause a fault in this mechanism – whether by flipping a bit in Flash memory [1780], or just finding the fuse and bridging it with two probing needles [303]. In other cases, the test routine is protected with a password, which can be found [1779].

We noted in section 5.7.1 that the RSA algorithm is fragile in the presence of failure; one laser shot is all it takes to cause a signature to be right modulo p and wrong modulo q, enabling the attacker to factor the key pq. Adi Shamir pointed out that if a CPU has an error in its multiply unit – even just a single computation $ab = c$ whose result is returned consistently wrong in a single bit – then you can design an RSA ciphertext for decryption (or an RSA plaintext for signature) so that the computation will be done correctly mod p but incorrectly mod q, again enabling you to factor the key [1708]. So a careful programmer will always check the results of critical computations, and think hard about what error messages might disclose.

How to hack a smartcard (6)

The next thing the pay-TV card industry tried was to incorporate hardware cryptographic processors, in order to force attackers to reconstruct hardware

circuits rather than simply clone software, and to force them to use more expensive processors in their pirate cards. In the first such implementation, the crypto processor was a separate chip packaged into the card, and it had an interesting protocol failure: it would always work out the key needed to decrypt the current video stream, and then pass it to the CPU which would decide whether or not to pass it on to the outside world. Hackers just tapped the wire between the two chips.

The next version had the crypto hardware built into the CPU itself. Where this consists of just a few thousand gates, an attacker can trace the circuit manually from micrographs. But with larger gate counts and deep submicron processes, a successful attack needs serious tools: you need to etch or grind away the layers of the chip, take electron micrographs, and use image processing software to reconstruct the circuit [270]. Equipment can now be rented and circuit-reconstruction software can be bought; the short resource now is skilled reverse engineers.

By the late 1990s, some pirates had started to get commercial reverse-engineering labs to reconstruct chips for them. Such labs get much of their business from analyzing integrated circuits on behalf of chip makers' competitors, looking for patent infringements. They also reverse chips used for accessory control, as doing this for compatibility rather than piracy is lawful. Many labs were located in Canada, where copying pay-TV cards wasn't a crime until 2002 (though there were at least two cases where these labs were sued by pay-TV operators). Some labs are now in China, whose legal system is harder for outsiders to navigate.

How to hack a smartcard (7)

In 1995 STM pioneered a new defence, a protective shield on the chip surface. This was a serpentine sensor line, zig-zagging round ground lines in a top metal layer. Any break or short would be sensed as soon as the chip was powered up, whereupon the chip would overwrite the keys.

Sensor mesh shields can really push up the cost of an attack. One bypass is to hold the sensor line to V_{DD} with a needle, but this can be fragile; and other vendors have multiple sensor lines with real signals on them. So if you cut them, you have to repair them, and the tool for the job is the *Focused Ion Beam Workstation* (FIB). This is a device similar to a scanning electron microscope but which uses a beam of ions instead of electrons. By varying the beam current, it can be used either as a microscope or as a milling machine, with a useful resolution under 10 nanometers. By introducing a gas that's broken down by the ion beam, you can lay down either conductors or insulators with a precision of a few tens of nanometers. For a detailed description of FIBs and other semiconductor test equipment that can be used in reverse engineering, see [1235].

FIBs are so useful in all sorts of applications, from semiconductor testing through metallurgy and forensics to nanotechnology, that they are widely

available in physics and material-science labs, and can be rented for about a hundred dollars an hour.

Given such a tool, it is straightforward to attack a shield that is not powered up. The direct approach is to drill a hole through the mesh to the metal line that carries the desired signal, fill it up with insulator, drill another hole through the center of the insulator, fill it with metal, and plate a contact on top – typically a platinum 'X' a few microns wide, which you then contact with a needle from your probing station (see Figure 18.5). There are many more tricks, such as using the voltage contrast and backscatter modes of your electron microscope to work out exactly where to cut, so you can disable a whole section of the mesh. John Walker has a video tutorial on how to use these tricks to defeat a shield at [1979].

Many other defenses can force the attacker to do more work. Some chips have protective coatings of silicon carbide or boron nitride, which can force the FIB operator to go slowly rather than damage the chip through a build-up of electrical charge. Chips with protective coatings are on display at the NSA Museum at Fort Meade, Maryland.

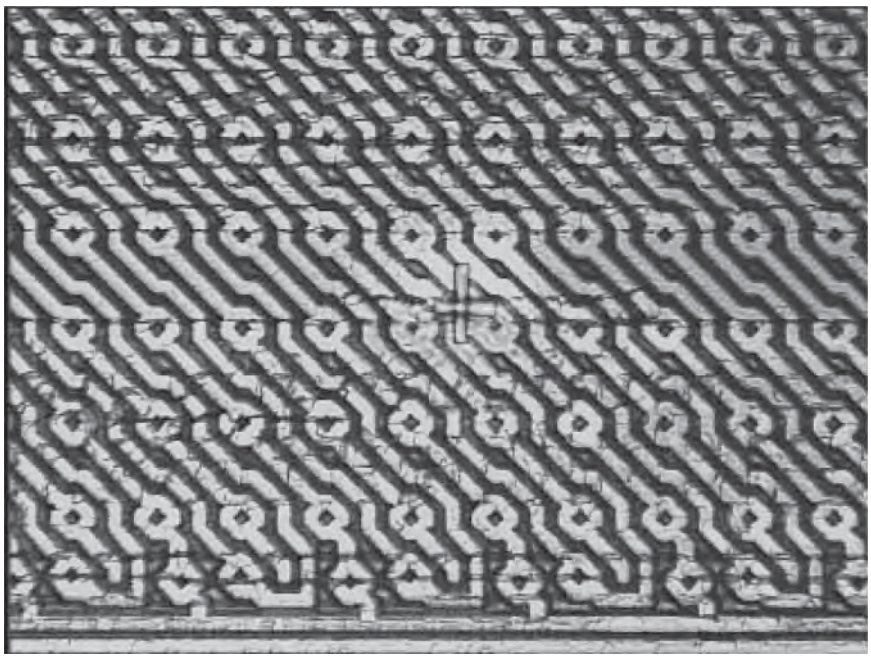

Figure 18.5: The protective mesh of an ST16 smartcard with a FIB cross for probing the bus line visible underneath (courtesy Oliver Kömmerling)

How to hack a smartcard (8)

In 1998, the smartcard industry was shaken when Paul Kocher announced a new attack known as *differential power analysis* (DPA). This relies on the fact

that different instructions consume different amounts of power, so by measuring the current drawn by a chip it was possible to extract the key. Smartcard makers had known since the 1980s that this was theoretically possible, and had even patented some crude countermeasures. But Paul came up with efficient signal processing techniques that made it easy, and which I'll describe in the following chapter. He came up with even simpler attacks based on timing; if cryptographic operations don't take the same number of clock cycles, this can leak key material too[4]. Power and timing attacks are examples of *side-channel attacks*, where the opponent can observe some extra information about the processor's state during a cryptographic computation. All the smartcards on the market in 1998 turned out to be highly vulnerable to DPA, and this held up the industry's development for a couple of years while countermeasures were developed.

Attacks were traditionally classed as either *invasive attacks* such as mechanical probing, which involves penetrating the passivation layer, and *noninvasive attacks* such as power analysis, which leaves the card untouched. Noninvasive attacks can be further classified into local attacks where the opponent needs access to the device, as with power analysis; and remote attacks where she could be anywhere, such as timing attacks. But that was not the whole story.

How to hack a smartcard (9)

Mechanical probing techniques have been getting steadily harder because of shrinking feature sizes. The next attack technology to develop was optical probing. The first report was from Sandia National Laboratories who in 1995 described a way to read out a voltage directly using a laser [33]. Since 2001 optical probing has been developed into an effective and low-cost technology, largely by my Cambridge colleague Sergei Skorobogatov. In 2002 Sergei and I reported using a photographic flashgun, mounted on the microscope of a probing station, to induce transient faults in selected transistors of an IC [1786]. The light ionises the silicon, causing transistors to conduct. Once you understand photoconductivity and learn to focus the light on single transistors, by upgrading from a flashgun to a laser, this enables many direct attacks. For example, microcontrollers can be opened by toggling the flip-flop that latches their protection state. This gave a new way of causing not just transient fault attacks, as on fragile cryptosystems such as RSA, but faults that are precisely directed and controlled in both space and time.

Later in 2002, Sergei reported using a laser mounted on the same cheap microscope to read out a microcontroller's memory directly. The basic idea is simple: if you shine a laser on a transistor, that will induce a photocurrent and increase

[4]On larger processors, it can be even worse; a number of researchers developed attacks on crypto algorithms such as AES based on cache misses during the 2000s, and in 2018 we had the Spectre and Meltdown attacks that exploit transient execution. See the chapter on side channels.

the device's power consumption – unless it was conducting already. So by scanning the laser across the device, you map which transistors are off and which are on. We developed this into a reasonably dependable way of reading out flip-flops and RAM memory [1651]. We named our attack *semi-invasive analysis* as it lies between the existing categories of invasive and non-invasive. It's not invasive, as we don't break the passivation; but we do remove the epoxy, so it doesn't count as non-invasive either.

Optical probing from the front side of the chip remained the state of the art for about five years. By the time of this book's second edition (2007), smartcard vendors were using 0.18 and 0.13 micron processes, typically with seven metal layers. Direct optical probe attacks from the chip surface had become difficult, not so much because of the feature size but because the metal layers get in the way. In addition, the sheer size and complexity of the chips was making it difficult to know where to aim. The difficulty was increased by *glue logic* – essentially randomised place-and-route.

Older chips have clearly distinguishable blocks, and quite a lot can be learned about their structure and organisation just by looking. Bus lines could be picked out and targeted for attack. However, the SX28 in Figure 18.6 just

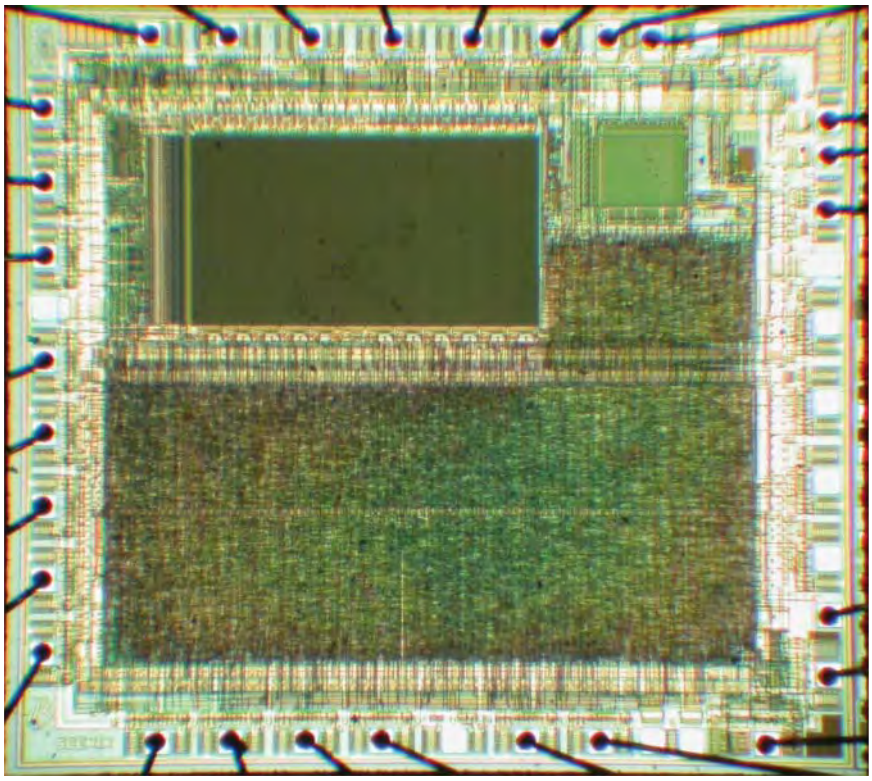

Figure 18.6: SX28 microcontroller with 'glue logic' (courtesy of Sergei Skorobogatov)

looks like a random sea of gates. The only easily distinguishable features are the EEPROM (at top left) and the RAM (at top right). It takes some work to find the CPU, the instruction decoder and the bus.

I wrote in the second edition, "The two current windows of vulnerability are the memory and the rear side." These have provided our Tamper Lab's main research targets during the decade since.

How to hack a smartcard (10)

Rear-side attacks are the practical semi-invasive option once you get below 0.35μ. You go through the back of the chip using an infrared laser at a wavelength around 1.1μ where silicon is transparent. For feature sizes below 65nm, you need to thin down the chip to 2–5μ using some combination of mechanical polishing and chemical etching; and there are now special methods to improve the resolution, such as silicon immersion lenses. One physical limit is you can't get a bandwidth of much over a few MHz because of the time taken for the charge carriers to recombine.

Rear-side attacks can sometimes be used to extract ROM contents by direct observation, but the main technique is optical fault induction (OFI), which has now become a standard security test procedure. Silicon immersion lenses have enabled OFI attacks to continue to create single-event upsets down to 28nm silicon, even though the laser spot size is about a micron [593]. Most smartcards current in 2019 tend to use about 90nm with the smallest about 65nm [1865]. The three big vendors have all announced 40nm products. So OFI will continue to be practical for some time.

With the smaller feature sizes, you have to accept that your aim in both space and time will often be fuzzy, and you may use the laser in combination with another more precise technique. One starting point here was optically-enhanced position-locked power analysis. By illuminating the n channels of a memory cell, the signal observed from a state change by power analysis is increased; with higher light levels, even read accesses can be detected. This enables much more selective analysis [1775].

How to hack a smartcard (11)

By 2010, the logic in most security chips was glue logic with few discernible features, but since Flash memory needs high voltages and large charge pumps, Flash arrays are large and easily identifiable. Chipmakers worried that the attacks that targeted chips with a separate V_{PP} programming voltage might be reinvented by using a laser to interfere with the charge pumps. So they tried to stop both memory corruption and the exploitation of memory readback access by making secure Flash with a per-block verify-only operation when memory is written. Sergei's *bumping attack* was inspired by the bumping attacks on locks described in Chapter 13. Just as lock bumping forces cylinders into a desired state, so Flash bumping forces bus lines into a desired state as they report the results of memory verification [1778].

But perhaps the most significant recent breakthrough was in 2016, when Franck Courbon, Sergei Skorobogatov and Chris Woods discovered how to use the latest generation of scanning electron microscopes to automate the direct read-out of Flash and EEPROM. As the memory cells store a bit by the presence or absence of a few hundred electrons in a floating gate, it's tricky to read them out without using the circuits designed for the purpose – especially when using a beam consisting of billions of electrons, aimed through the rear side of the chip. (We used to compare this with reading a palimpsest with a blowlamp.) Making it work requires very careful sample preparation, a SEM that supports passive voltage contrast (PVC), fine-tuned scan acquisition and efficient image processing [480]. Using such tools and techniques, it's now possible to read out the 256K of Flash or EEPROM from a typical smartcard or other security chip with perhaps half a dozen single-bit errors. This had been predicted as long ago as 2000 by Steve Weingart, the 4758 designer [2003]; PVC made it a reality. The effect on the smartcard industry is that the entire memory of the chip can now be read out. Reverse engineering is a matter of figuring out the CPU's instruction set, how the memory is encrypted, and so on.

How to hack a smartcard (12)

Reverse engineering services in China now charge 30c per gate, so the brute-force approach is to just reverse the whole chip and drop it in a simulator without trying to understand it in detail. Given that a typical smartcard has 100,000 gates, this means you can get a simulator for $30,000. Then you have all sorts of options. Once you have sufficiently understood one card of a particular type, the per-card cloning cost is now the cost of memory extraction. You can also use the simulation to look for side-channel attacks, to plan FIB edits, or to fuzz the device and look for other vulnerabilities.

As smartcards are computers, they can sometimes fall to the usual computer attacks, such as stack overwriting, by sending too long a string of parameters. As early as 1996, the Mondex card, used in a payment trial by the UK's NatWest Bank, boasted a formally verified operating system. Yet as late as 2019, software attacks worked against at least one SIM card. Malicious SMSes were used by nation-state attackers to download malware into the SIMs of target users so that their location could be tracked [575].

18.5.4 Random number generators and PUFs

Many crypto chips are offered with a random number generator, a physical unclonable function, or both.

Hardware random number generators (RNGs) are used to produce protocol nonces and session keys. Weak generators have led to many catastrophic security failures, of which a number pop up in this book. Poor nonces lead

to replay attacks, while weak session keys can compromise long-term signing keys in cryptographic algorithms such as ECDSA. During the 1990s, the fashion was for algorithmic random number generation; this is properly known as a *pseudorandom number generator* (PRNG). A crypto chip might have had a special key-generation key that was used in counter encryption mode; operating systems often had something similar. However, if the counter is reset, then the output is repeated; there have been several variants on this theme. I also mentioned the NIST Dual-EC-DRBG, which was built into Windows and seemed to have contained an NSA trapdoor [1737]; Ed Snowden later confirmed that the NSA paid RSA $10m to use this standard in tools that many tech companies licensed [1292].

Hardware random number generators typically quantise jitter or use some source of metastability such as a cross-coupled inverter pair. Such generators are notoriously difficult to test; faults can be induced by external noise such as temperature, supply voltage and radiation. Standards such as NIST SP800-A/B/C call for RNG output to be run through test circuits. Crypto products often mix together the randomness from a number of sources both environmental and internal [839], and this is a requirement for the highest levels of certification. The way these sources are combined is often the critical thing and one should beware of designs that try to be too clever [1035]. One must also beware that hardware RNGs are usually proprietary, obscure designs, sometimes specific to a single fab, so it's hard to check that the design is sound, let alone that it doesn't contain a subtle backdoor. An example of conservative design may be that used in Intel chips since 2012, which combines both a hardware RNG and a software PRNG that follows it [857].

The manufacture of crypto chips typically involves a personalisation stage where serial numbers and crypto keys are loaded into Flash or EEPROM. This is another attack point: Ed Snowden reported that GCHQ had hacked the mechanisms used by Gemalto to personalise cards, and got copies of the keys in millions of SIMs. So one might ask whether chips could be manufactured with an intrinsic key that would never leave the device. Each chip would create a private key and export the public key, which the vendor would certify during personalisation. But this takes time, and also seems to need an RNG on the chip. Is there another way?

A physical unclonable function (PUF) is a means of identifying a device from variations that occur naturally during manufacture. In the 1980s, Sandia National Laboratories were asked by the US Federal Reserve whether it was possible to make unforgeable banknote paper, and they came up with the idea of chopping up optical fibre into the mash, so you could recognise each note by a unique speckle pattern [1750]. Such a mechanism should be unclonable, and its behaviour should change detectably if it's tampered with. Could something similar be devised for integrated circuits? In 2000, Oliver

and Fritz Kömmerling proposed loading chip packaging with metal fibres and measuring its properties to generate a key with which the chip contents would be encrypted, so that drilling through the packaging would destroy the key [1081]. In 2002 Blaise Gassend, Dwaine Clarke, Marten Van Dijk and Srini Devadas proposed using process variability in the silicon itself, suggesting that a collection of ring oscillators might be chaotic enough to be unique [754]. There followed the usual coevolution of attack and defence as people proposed designs and others broke them.

Through the 2010s we've started to see PUFs appearing in significant numbers of low-cost chips as well as in higher-value products such as FPGAs. The typical 'weak PUF' generates a consistent random number on power-up from process variability; an SRAM PUF reads the initial state of some SRAM cells and is used, with error correction, as a stable random ID or as an AES key to encrypt memory or to drive a PRNG. If your opponent is capable of reversing your circuit and scanning your Flash memory, a PUF may at least force them to go to the trouble of probing the key off the bus, or inducing faults one bus line at a time to read it out using differential fault analysis.

PUF marketing often claims much more, and one claim (as well as a research goal) is a 'strong PUF' which would act as a hardware challenge-response mechanism. Given an input, it would return an output that would be sufficiently different for each chip (and each input) to be usable as a cryptographic primitive in itself. For example, one might send a thousand challenges to the chip at personalisation and store the responses for later key updating. Note that this would not of itself have stopped the NSA attack on Gemalto, as they hacked the personalisation files and if PUFs had been used they'd have got the challenge-response pair files too.

The state of the art in 2020 appears to be *XOR arbiter PUFs*, which consist of a chain of multiplexers followed by an arbiter. The challenge to the PUF is input to the address lines of the multiplexers that select a route for signals to race through them to get to the arbiter. To make it harder for an attacker to work out the relative delay on each circuit path, the outputs of a number of arbiters are XORed together. However, Fatemeh Ganji, Shahin Tajik and Jean-Pierre Seifert have shown that suitable machine-learning techniques can be used to model the underlying circuits [745]. The same authors worked with Heiko Lohrke and Christian Boit to develop laser fault induction attacks, guided by the chip's optical emissions, that disable some arbiters so that others can be learned more quickly, and thus significantly reduce the PUFs' entropy [1862]. There are always probing attacks, as some routine on the chip has to be able to read the PUF for it to do any work, and this means the bootloader or the monitor. As these are often left open to parts of the supply chain for personalisation, warranty and upgrade purposes, it's hard to see what extra protection such devices would give, even if we could invent one that works properly. Also, using such devices at scale would tend to make personalisation slower and

protocols more complex. Finally, the strength of a PUF depends on variation that the fab tries its best to eliminate, so a change in silicon process can suddenly make a PUF design insecure.

18.5.5 Larger chips

There's a growing number of larger chips with embedded security functions, typically aimed at manufacturing control or accessory control. The granddaddy of these products may be the *Clipper chip*, which the Clinton administration proposed in 1993 as a replacement for DES. Also known as the *Escrowed Encryption Standard* (EES), this was a tamper-resistant chip containing the Skipjack block cipher and a protocol designed to allow the FBI to decrypt any traffic encrypted using it. When you gave Clipper some plaintext and a key to encrypt it, the chip returned not just the ciphertext but also a *Law Enforcement Access Field* (LEAF), which contained the user-supplied key encrypted under an FBI key embedded in the device. To prevent people cheating by sending the wrong LEAF with a message, the LEAF had a MAC computed with a 'family key' shared by all Clipper chips – which had to be tamper-resistant to keep both the Skipjack block cipher and the LEAF family key secret.

As often happens, it wasn't the tamper-resistance that failed, but the protocol. Almost as soon as Clipper hit the market, Matt Blaze found a vulnerability: as the MAC used to bind the LEAF to the message was only 16 bits long, it was possible to feed message keys into the device until you got one with a given LEAF, so a message could be sent with a LEAF that would reveal nothing to the government [259]. Clipper was replaced with the Capstone chip, the crypto wars continued by other means, and the Skipjack block cipher was placed in the public domain [1402].

Of interest in this chapter are the tamper protection mechanisms used, which were claimed at the time to be sufficient to withstand a 'very sophisticated, well funded adversary' [1400]. Although it was claimed that the Clipper chip would be unclassified and exportable, I was never able to get hold of a sample despite repeated attempts. It used *Vialink read only memory* (VROM) in which bits are set by blowing antifuses between the metal 1 and metal 2 layers on the chip. A high-voltage programming pulse is used to melt a conducting path through the polysilicon between two metal layers. This technology was also used in the QuickLogic FPGA, which was advertised as a way for firms to conceal proprietary algorithms, and claimed to be 'virtually impossible to reverse engineer'; further details and micrographs appeared in its data book [802]. A recent variant is the *spot breakdown PUF* where a high enough voltage is applied to a bank of transistors for just long enough that about half of them suffer breakdown of the gate oxide, creating random failures that can be read as ones and zeros [424].

Fusible links are used on other devices too; recent iPhones, for example, have an AES key burned into the system-on-chip. There are basically three approaches to reverse engineering an antifuse device.

- The first thing to look at is the programming circuitry. All such chips have a test circuit used to read back and verify the bitstream during programming, and many disabled this by melting a single fuse afterwards. If you can get sample devices and a programmer, you can maybe find this fuse using differential optical probing [1776]. You then use a FIB to repair it, or bridge it with two probe needles, and read out the bitstream. This attack technique works not just for antifuse FPGAs but also for the Flash and EEPROM varieties.
- Where you need to read out many fuses, as where they're used to store an AES key, the brute-force approach is to strip the chip down one layer at a time and read the fuses directly; they turn out to be visible under a suitable chemical stain. As this attack is destructive it is typically of limited interest against keys that are different in each device (as in the iPhone, or a spot breakdown PUF).
- Where the device implements a cryptographic algorithm, a side-channel attack may be the fastest way in. Most devices manufactured before about 2000 are rather vulnerable to power analysis, and while smartcard chipmakers have incorporated defences, the makers of larger chips may have preferred to avoid paying royalties to Cryptography Research, which patented many of the best ones. You can always try optical fault induction to read the key one bit at a time, and since the late 2000s we also know how to work with optical emissions, which I'll discuss later.

Secure FPGAs became big business in the 21st century as firms outsource the manufacture of electronic goods to the Far East but want to control at least one critical component to prevent overbuild and counterfeiting. Most FPGAs sold now have conventional memory rather than antifuse, so they can be made reprogrammable. If you use a volatile FPGA that stores the bitstream in SRAM, you will want one or more embedded keys kept in nonvolatile memory, so the bitstream is uploaded and then decrypted on power-up. For faster power-up you might choose a non-volatile device that stores the whole bitstream in Flash. In both cases, there may be fuses to protect the key material and the security state [583]. But do watch out for service-denial attacks via the upgrade mechanism. For example, a Flash FPGA may only have enough memory for one copy of the bitstream, not two; so the naïve approach is to read in the bitstream once to decrypt it and verify the MAC, and then a second time to reprogram the part. But if the bitstream supplied the second time is corrupt, will you have a dead product? And if you allow rollback, your customers can perhaps escape upgrades by replaying old bitstreams. And if an attacker gets your products to

load a random encrypted bitstream, this could cause short circuits and brick the part. So stop and think whether anyone might try to destroy your product base via a corrupt upgrade; if so, you might consider a secure bitstream loader. You might also consider a more expensive FPGA with enough on-chip memory to support old and new bitstreams at the same time.

The second type of large-chip security product is the *system-on-chip* (SoC) with inbuilt authentication logic. The pioneer may have been Sony's Playstation 2 in 2000, which fielded MagicGate, a cryptographic challenge-response protocol run between the device's graphics chip and small authentication chips embedded in legitimate accessories. The business model of games console manufacturers included charging premium prices for software and additional memory cards, whose sellers had to use copy-control technology and pay the console vendor a royalty; this was used to subsidise the initial cost of the console. Of course, aftermarket operators would then hack their copy-control mechanisms, so Sony set out to dominate its aftermarket with a better copy-control technology. This used some interesting protection tricks; the MagicGate protocol was both simple (so protocol attacks couldn't be found) and randomised (so attackers couldn't learn anything from repeating transactions). It took several years and millions of dollars for the aftermarket firms to catch up. While the authentication logic in a small chip may need a top metal shield, copy traps and layout obfuscation to hide it, the same logic in a large chip can hide among the billions of other transistors.

By the mid-to-late 2000s, similar logic was appearing in system-on-chip products in other industries – sometimes for accessory control, and sometimes to enable one product to be sold with several different levels of performance as a means of price discrimination. This practice has led to some interesting edge cases. For example, in 2017 Tesla temporarily 'upgraded' the batteries of its model S and X cars so that owners could get out of the path of Hurricane Irma [1934].

So how can you hack the magic devices that we find everywhere nowadays? Memory readout can be the most dependable attack path. As an example, Sergei Skorobogatov used the new PVC Flash / EEPROM readout technique to reverse the OmniPod insulin pump. Diabetics who know how to program prefer to control their own insulin pumps but vendors try to stop them, for both market control and liability reasons. The OmniPod's system-on-chip therefore runs an authentication protocol with the device's authorised controller, and the Nightscout Foundation, an NGO that supports diabetics, wanted to extract the keys so patients could optimise the control for their own health needs rather than following the treatment protocols devised by Omnipod. The analysis is described in [1782].

A second attack path is to look to see whether the device computes with encrypted data, and if so look for a protocol failure or side channel that gives a way in. An early example was the *cipher instruction search attack* invented

by Markus Kuhn on the DS5002 processor [1104]. This device pioneered *bus encryption* with hardware that encrypts memory addresses and contents on the fly as data are loaded and stored, so it was not limited to the small amount of RAM that could be fitted into a low-cost tamper-sensing package at the time (1995). Markus noticed that some of the processor's instructions have a visible external effect; one instruction in particular caused the next byte in memory to be output to the device's parallel port. So if you intercept the bus between the processor and memory using a test clip, you can feed in all possible 8-bit instruction bytes at some point in the instruction stream until you see a one-byte output. After using this technique to tabulate the encryption function for a few bytes, you can encipher and execute a short program to dump the entire memory contents. Similar tricks are still used today, and variants on the attack still work. In 2017 Sergei Skorobogatov demonstrated an active attack on a system-on-chip used in the car industry, which used memory encryption to make bus probing harder. By selectively injecting wrong opcodes into the bus, he was able to reverse the encryption function [1783].

A tougher problem was presented by the iPhone. In March 2016 FBI director James Comey demanded that Apple produce a law-enforcement 'upgrade' to its iOS operating system to enable access to locked iPhones, claiming that the FBI would otherwise be unable to unlock the phone of the San Bernardino shooter. Sergei set out to prove him wrong and by August had a working attack. The phone in question, the Apple 5c, has an SoC with an embedded AES key, set up by burning fusible links; as these can be seen under an electron microscope, read-out may be possible but would destroy the SoC. AES isn't vulnerable to cryptanalysis, and the encryption appears to work one cache line at a time, so cipher instruction search won't work. But no matter, as there's a *NAND mirroring attack*. The phone's non-volatile memory is a NAND Flash chip whose contents are encrypted, one cache line at a time, by the embedded device key, so that the chip from one phone can't be read in another. The attack is to desolder the memory chip, mount it in a socket, and copy its contents. You then make half a dozen PIN guesses, and the phone starts to slow down (it locks after ten). Next, you remove the memory chip and restore its original contents. You can now make half a dozen more attempts. With a bit more work, you can clone the chip or build a circuit board to emulate it, so you can guess faster. The details can be found in [1781]. In the end, the FBI used a service from Cellebrite, a forensics company, which later turned out to be exploiting the Checkm8 bug in the iPhone ROM [794].

The third type of attack I'll mention is *optical emission analysis*, which is strictly speaking a side channel but which I'll introduce here as it's becoming one of the main ways of attacking high-grade crypto chips. Photons are emitted when semiconductor junctions switch, and photon emission microscopy is an established failure analysis technique, with silicon emitting mostly in the near infrared near the drain area of n-MOS transistors. This was first used to attack a

crypto implementation in 2008 by Julie Ferrigno and Martin Hlavac, who used an expensive single-photon counting photomultiplier to read out AES keys from an outdated 0.8μ microcontroller, but worried that their technique would not work for technologies smaller than 0.12μ [681]. By the following year, Sergei Skorobogatov found that a photomultiplier sold to hobby astronomers was near ideal and discovered a voltage boost trick: increasing the chip supply voltage from 1.5V to 2V increases the photon output sixfold. He found he was almost able to read out the AES keys from the internal crypto engine of a modern chip, the Actel ProASIC3 FGPA. Then, once the AES algorithm timing had been established, and he knew each round key took 1.6μs, he further increased the voltage to 2.5V for the 0.2μs of an individual bus write, giving a further fourfold increase in the photon output plus temporal resolution, which enabled him to read each word of round key clearly off the bus. This was all rather embarrassing as I'd consulted on the design to Actel back in 2001. The ProASIC3 was fabricated in a 0.13μ technology with 7 metal layers and flash memory, and we had built in all sorts of countermeasures to block the attacks we knew about at the time; reading it out invasively would have been tedious. That was a sharp reminder that it's hard to block the attacks that haven't been invented yet, and that attacks can improve very quickly once experts start to hone them. Optical emission analysis is now used in combination attacks: if you want to attack a chip that's too big to reverse engineer, you observe the emissions as it does the cryptography and this tells you where to aim your laser as you try a fault attack or optically-enhanced power analysis. It can also suggest where you might lay down a few probe points with your FIB.

18.5.6 The state of the art

How well can you protect a single-chip product against a capable motivated opponent? In the late 1990s, everything got broken, and in the 2001 edition of this book, I wrote, "there isn't any technology, or combination of technologies, known to me which can make a smartcard resistant to penetration by a skilled and determined attacker." During the 2000s, the defence improved because of the efforts of the pay-TV firms and the banking industry, so in the second edition I wrote "This is still almost true, but … you can be looking at a year's delay, a budget of over a million dollars, and no certainty of success."

Now, in 2019, Moore's law has run out of steam; crypto chips are mostly stuck at about 100nm, while the semiconductor test equipment industry is aiming to support 9nm processing and still turning out innovations such as passive voltage contrast microscopy; and researchers are finding innovative ways to use their products. So the attackers are starting to catch up. The scope of the industry is also increasing. In 2007, we had a handful of smartcard OEMs, a handful of reversing labs and a handful of interested academics; now many chipmakers are being asked by their customers for some tamper-resistance, as products

from routers to the Raspberry Pi acquire some kind of secure boot capability to defeat persistent malware. So there are ever more medium-grade products that are suitable for grad students to learn the art and craft of hardware reverse engineering[5]. And the growing demand, particularly in China, to reverse devices for compatibility drives the growth of commercial reversing labs. The market is now big enough for people to make a living selling specialist tools such as layout-reconstruction software and optical fault induction workstations. As a result, attackers are getting more numerous and more efficient. I suspect that the cost of cloning a smartcard will steadily come down through the tens of thousands and perhaps into the single thousands.

Security economics remains a big soft spot, with security chips being in many ways a market for lemons. A banker buying HSMs probably won't be aware of the huge gap between FIPS level 3 and level 4, and understand that level 3 can sometimes be defeated with a Swiss army knife. The buying incentive there is compliance, and where real security clashes with operations it's not surprising to see weaker standards designed to make compliance easier. API security is too hard, and the difference between HSMs' internal and external APIs makes it too confusing. The near-abdication of FIPS in favour of ISO 19790 and various protection profiles touted under the Common Criteria will confuse things further, as will the UK's move away from the Criteria. Confusion marketing and liability games appear set to continue. But does this matter?

First, most of the HSM business is moving to the cloud, with Azure and AWS each having of the order of 2,000 HSMs, and Google playing catchup. Instead of having a few thousand banks each running a few, or a few dozen, HSMs we'll have three companies running a few thousand. As the prices are driven down, the HSM vendor engineers' expertise will be lost; and as the cloud service providers guard their datacentres, HSMs are likely to be replaced by crypto chips.

Second, most of the volume smartcard markets – SIM cards and EMV cards – have only moderate physical protection requirements as a full compromise enables the attacker to exploit one account only. You don't want a bad terminal to be able to do production power-analysis attacks on every EMV card it sees, but even if that were to happen it's not the end of the world, as that's how mag-stripe cards got cloned, and we know how to limit the damage. The pay-TV markets used to lead innovation and customise the chips they used, as a single break can enable a pirate to sell hundreds of thousands of clone cards. But pay-TV is now moving to wireline broadband, and the companies learned that more secure chips aren't the only way to cut losses: more complex smartcards played a role, but much of the improvement came from legal action against pirates, and from making technical and legal measures

[5]My colleagues Franck Courbon, Markus Kuhn and Sergei Skorobogatov now run just such a course for our graduate students.

work together efficiently. Gadget makers nowadays lock their products into ecosystems with cloud services and apps, which makes manufacturing control less dependent on tamper-proof FPGAs.

I therefore expect that although the number and variety of crypto chips will continue to increase, the quality of physical protection will remain indifferent. Vendors will spend only as much money as they need to in order to meet certification requirements, which will remain slippery and will be gamed. Security engineers will have to get used to building systems out of grey-box components – chips from which keys and algorithms can be extracted, given some effort.

I suspect that accessory control will remain the toughest hardware battlefield. Aftermarket control isn't just about printer cartridges nowadays but extends to vehicles, medical devices and other high-value products. But where at least one of the two devices that authenticate each other goes online at least occasionally, the protection requirements are much less severe than for satellite TV. The real question will be how to stop attacks scaling.

18.6 The residual risk

The security engineer will therefore have to pay attention to the many failure modes of systems involving tamper-resistant processors that are more or less independent of the price or technical tamper-resistance of the device.

18.6.1 The trusted interface problem

None of the devices described in the above sections has a really trustworthy user interface[6]. Some of the bank security modules have a physical lock (or two) on the front to ensure that only the person with a given metal key (or smartcard) can perform privileged transactions. But whether you use a $2000 4765 or a $2 smartcard to do digital signatures, you still trust the PC that drives them. If it shows you a text saying "Please pay amazon.com $67.99 for a copy of Anderson's *Security Engineering*" while the message it actually sends for signature is "Please remortgage my house at 13 Acacia Avenue and pay the proceeds to Mafia Real Estate Inc", then the tamper resistance hasn't bought you much.

Indeed, it probably makes your situation worse. Nick Bohm, Ian Brown and Brian Gladman pointed out that when you use a qualifying electronic signature device, you're saying 'I agree to be unreservedly liable for all signatures that are verified by the key that I now present to you and I will underwrite all

[6]The iPhone secure enclave processor (SEP) has a direct link to the fingerprint reader but relies on the main application processor for everything else including FaceID.

the risks taken by anyone as a result of relying on it' [278]. I will discuss the history and politics of this later in section 26.5.2. The EU eIDAS regulation requires all EU governments to accept qualifying electronic signatures for transactions where they previously required ink on paper, and set standards for technical certification of signature devices. The industry has duly produced dozens of certified products. Given the liability shift compared with ink-on-paper signatures, no sensible person would use a qualifying electronic signature device unless they had to. So the lobbyists have been at work, and some countries now insist you use them to file your taxes. This has led researchers in Germany to look closely at how signatures, signature verification services and PDF files interact; as you might expect, the results are somewhat shocking. Vladislav Mladenov, Christian Mainka, Karsten Meyer zu Selhausen, Martin Grothe and Jörg Schwenk created a document signed by Amazon in Germany and backed by all the official machinery, certifying that you are due a refund of one trillion dollars. They found three new attacks on pdf signatures, worked out how to bypass signature validation in 21 out of 22 viewers, and cheated 6 of 8 online validation services [1328]. It's a fair bet that this is just the tip of an iceberg.

Another example comes from the hardware wallets that some people use to store cryptocurrency. Early products had no trusted display and were thus vulnerable to malware. Some later ones combined a smartcard chip acting as a secure element, with a less secure microcontroller driving a display. This opens a number of possibilities – including an *evil maid attack* described by Saleem Rashid where someone with temporary access to the device, such as a hotel maid, reflashes the microcontroller software [1583]. In this case the secure element had no idea whether the main processor was running compromised code.

Trustworthy interfaces aren't always needed, as tamper-resistant processors are often able to do useful work without having to authenticate users. Recall the example of prepayment electricity metering in section 14.2: there, tamper-resistant processors can maintain a value counter, enforcing a credit limit on each operator and limiting the loss when a vending machine is stolen. Postal meters work the same way. In many other applications from printer ink cartridges through games consoles to prepaid phone cards, the vendor mainly cares about use control.

18.6.2 Conflicts

A further set of issues is that where an application is implemented on devices under the control of different parties, you have to consider what happens when each party attacks the others. In banking, the card issuer, the terminal owner and the customer are different; all the interactions of cloned cards, bogus terminals, gangland merchants and cheating banks need to be thought through.

A particular source of conflict and vulnerability is that many of the users of tamper resistance have business models that make their customers the

enemy – such as rights management and accessory control. Their customers may own the product, but have the incentive to tamper with it if they can. In the case of accessory control, they may also have a legal right to try to break it; and where the mechanisms are used to limit device lifetime and thus contribute to environmental pollution, they may even feel they have a moral duty.

18.6.3 The lemons market, risk dumping and evaluation games

Each of the product categories discussed here, from HSMs down through FPGAs to smartcards, has a wide range of offerings with wide variability in the quality of protection. Many products have evaluations, but interpreting them is hard.

First, there are relatively few offerings at high levels of assurance – whether FIPS-140 level 4 or Common Criteria levels above 4. There are many at lower levels, where the tests are fairly easy to pass, and where vendors can shop around for a lab that will give them an easy ride. This leads to a lemons market in which all but the best informed buyers will be tempted to go for the cheapest FIPS level 3 or CC EAL4 product.

Second, evaluation certificates don't mean what they seem. Someone buying a 4758 in 2001 might have interpreted its level 4 evaluation to mean that it was unbreakable – and then been startled when we broke it. In fact, the FIPS certificate referred only to the hardware, and we found vulnerabilities in the software. It's happened the other way too: there's been a smartcard with a Common Criteria level 6 evaluation, but that referred only to the operating system – which ran on a chip with no real defences against microprobing. I'll discuss the failings of evaluation systems at greater length in Part 3.

Third, while HSMs tend to be evaluated under FIPS, smartcard vendors tend to use the Common Criteria. There the tussles are about which protection profile to use; vendors naturally want the labs to evaluate the aspects of security they think they're good at.

Finally, many firms use secure processors to dump risk rather than minimise it. Banks love to be able to say 'your chip and PIN card was used, so it's your fault' and in many countries the regulators let them get away with it. There are many environments, from medicine to defense, where buyers want a certificate of security rather than real protection, and this interacts in many ways with the flaws in the evaluation system. Indeed, the main users of evaluated products are precisely those system operators whose focus is on due diligence rather than risk reduction.

18.6.4 Security-by-obscurity

Many designers have tried hard to keep their cryptoprocessor secret. You have almost always had to sign an NDA to get smartcard development tools. The

protection profiles still used for evaluating many smartcards under the Common Criteria emphasise design obscurity. Chip masks have to be secret, instruction set architectures are proprietary, staff have to be vetted, developers have to sign NDAs – these all pushed up industry's costs [656]. Obscurity was also a common requirement for export approval, leading to a suspicion that it covers up deliberate vulnerabilities. For example, a card we tested would always produce the same value when instructed to generate a private/public keypair and output the public part. Many products that incorporate encryption have been broken because their random number generators weren't random enough [576,776] and as we discussed, the NSA got NIST to standardise a weak one.

Some HSM vendors have been an honourable exception; IBM's Common Cryptographic Architecture has been well documented from the beginning, as has Intel's SGX and the core mechanisms of Arm's TrustZone. This openness has facilitated the discovery of API attacks on IBM's product, as well as side-channel and ROP attacks on Intel's and more recently Arm's. But most such attacks have been disclosed responsibly and the learning process has improved their products.

One tussle in 2020 is over whether the development environment needs to be air-gapped. This has been common practice for years in smartcard OEMs; one lab we visited had only a single PC connected to the Internet (painted red, on a pedestal) so staff could book flights and hotels. These firms are now pushing evaluators to emphasise the risk that an attacker ends up owning the entire company infrastructure using an advanced persistent threat. That would make life inconvenient for firms that have always operated online, as they would have to rebuild toolchains and change their workflows.

A smart evaluator would not be taken in by such gamesmanship. Almost none of the actual attacks on smartcards used inside information; most of them started out with a probing attack or side-channel attack on a card bought at retail. As the industry did not do hostile attacks on its own products in the early years, its products were weak and were eventually broken by others. Since the late 1990s some organisations, such as VISA, have specified penetration testing [1967]. But the incentives are still wrong; a sensible vendor will go to whatever evaluation lab offers the easiest ride. We'll discuss the underlying economics and politics of evaluation in Section 28.2.7.2.

18.6.5 Changing environments

We've already seen examples of how function creep and changes in the environment have broken systems by undermining their design assumptions. A general problem is 'leverage' – where firms try to exploit infrastructure maintained by others, without negotiating proper contracts. We've seen how the SIM card that was previously just a means of identifying people to the phone company became a token that controls access to their bank accounts. In the second edition of this book, I wrote "Does this matter? … I'd say it

probably doesn't; using text messages to confirm bank transactions gives a valuable second authentication channel at almost zero marginal cost to both the bank and the customer." At that time, we had one reported case of a SIM swap attack, in South Africa.

In the following paragraph, I wrote: "But what will happen in five or ten years' time, once everyone is doing it? What if the iPhone takes off as Apple hopes, so that everyone uses an iPhone not just as their phone, but as their web browser? All of a sudden the two authentication channels have shrunk to one." And so it is; SIM swap is now going mainstream in the USA.

This is actually tied up with local law and regulation. In most countries, phone companies are not liable to banks for failing to authenticate their customers properly. After all, phone companies just sell minutes, and the marginal cost of stolen minutes is near zero. But one country with little SIM swap fraud is India, where regulators decided that phone companies must share liability for SIM swap fraud, and where the phone company is required to check a customer's fingerprint against the national Aadhar database before selling them a SIM.

18.7 So what should one protect?

In such a complex world, what value can tamper-resistant chips add?

First, they can tie information processing to a single physical token. A pay-TV subscriber card can be bought and sold in a grey market, but so long as it isn't copied the station operator isn't losing much revenue. This also applies to accessory control, where a printer vendor wants their product to work with any genuine ink cartridge, just not with a cheap competitor.

Second, they can maintain value counters, as with the postal metering discussed in Chapter 13. Even if the device is stolen, the total value of the service it can vend is limited. In printers, ink cartridges can be programmed to dispense only so much ink and then declare themselves dry.

Third, they can reduce the need to trust human operators. Their main purpose in some government systems was 'reducing the street value of key material to zero'. A crypto ignition key for a secure phone should allow a thief only to masquerade as the rightful owner, and only if they have access to an actual device, and only so long as neither the key nor the phone has been reported stolen. The same general considerations applied in ATM networks, which not only implement a separation-of-duty policy, but transfer a lot of the trust from people to things.

Fourth, they can protect a physical root of trust that monitors secure boot, and thus make it hard for malware to be persistent. This mission of its own

does not require high-grade physical protection; security against capable motivated software attackers is the key. One question is whether activists who want to run their own favoured version of Linux on their devices actually have to break the TPM, or whether they can just ignore it and manage the malware risk themselves.

Fifth, they can control the risk of overproduction by untrusted hardware contractors: sometimes called the 'third shift' problem, where the factory you hire runs two shifts to make devices for you and a third shift to make some more for grey-market sale. This can involve embedding part of the design in an FPGA that's hard to reverse engineer, or by having a TPM to control the credentials necessary for the device to work in your ecosystem. As things acquire cloud services and apps, firms are moving from the former strategy to the latter, which has lower hardware costs and is easier to manage. You just release as many credentials as the factory ships you products.

Sixth, such techniques can control some of the more general risk from counterfeit electronic parts. This covers a multitude of sins, from cheap knock-offs that cause early product failure through to sophisticated supply-chain attacks by state adversaries. For a survey, see Guin et al [834]. The techniques described in this chapter also find use in the fight against counterfeiting, as do many of the tools. As for supply-chain attacks, the most pernicious may be hardware trojans. One national-security concern is that as defence systems increasingly depend on chips fabricated overseas, the fabs might introduce extra circuitry to facilitate later attack. For example, some extra logic might cause a 64-bit multiply with two specific inputs to function as a kill switch. This has been the subject of significant research since about 2010, and mechanisms have been developed for Trojan detection both pre-silicon and post-silicon; for example, you can do a differential side-channel analysis of a 'golden' reference chip and a target of evaluation [1779]. This of course assumes that you can produce a reference chip in a trustworthy fab. For a survey of this field, see Xiao et al [2056].

This is an incomplete list. But what these applications have in common is that a security property can be provided independently of the trustworthiness of the surrounding environment. But beware: the actual protection properties that are required and provided can be quite subtle, and tamper-resistant devices are more often a useful component than a full solution. Generic mechanisms fail again and again; security is not some kind of magic pixie dust that you sprinkle on a system to cause bad things to not happen. You need to work out what bad things you want to stop. If you're not careful you can find yourself paying for smartcards and crypto modules in applications where they add rather little; and if you're really unlucky you may find that the industry lobbied for legal mandates or industry standards to force you to use their products.

18.8 Summary

Tamper-resistant devices and systems have a long history. Computers can be protected against physical tampering in a number of ways, from keeping them locked up in a guarded room, through putting them in tamper-sensing boxes, to making them into single chips with shields against probing and defences against side-channel attacks.

I've told the story of how hardware tamper-resistance developed through a series of cycles of attack and defence, and given examples of applications. Security processors are typically vulnerable to attacks on interfaces (human, sensor or system) but can often deliver value in applications where we need to link processing to physical objects and to protect security state against scalable threats, particularly in environments where any online service may be intermittent.

Research problems

There are basically two strands of research in tamper-resistant processor design. The first concerns itself with making 'faster, better, cheaper, more secure' processors: how can the protection offered by a high-end device be brought to chips that cost under a dollar? The second concerns itself with pushing forward the state of the attack art. How can the latest chip testing technologies be used to make 'faster, better, cheaper, novel' attacks? The best guide for the second may be Sergei Skorobogatov's 2018 talk, "Hardware Security: Present challenges and Future directions" [1784].

A broader area of research is how to build more secure systems out of less secure components. How can moderately protected chips be used effectively to stop various kinds of attack scaling?

Further reading

I'm not aware of any up-to-date systematisation of knowledge paper on hardware tamper resistance. Colleagues and I wrote a survey of security processors in 2005 [101], which might serve as a more detailed starting point, if slightly dated; of the same vintage are a summer school on attack techniques [1776] as well as reviews of FPGA security [583] and microcontroller security [1771,1773]. Bunnie Huang's book on hacking the Xbox is still a good read [932]. A slightly later summary, from an industry perspective, is by Randy Torrance and Dick James of Chipworks in 2009 [1901].

As for the last decade of research, the best current papers often appear at conferences such as CHES (for the crypto), HOST (Trojans and backdoors), FDTC (fault attacks) and Cardis (smartcards). Failure analysis research tends to appear at ISTFA and IPFA.

For the early history – the weighted codebooks and water-soluble inks – read David Kahn's book '*The Code Breakers*' [1003]. For a handbook on the chip card technology of the mid-to-late 1990s, see [1581], while the gory details of how we started tampering with those generations of cards can be found in [107,108,1080]. The IBM products mentioned have extensive documentation online [953], where you can also find the US FIPS documents [1399].

For modern chip testing techniques, I recommend the video of a keynote talk by John Walker at Hardwear.IO 2019 on how to use FIBs in reverse engineering [1979] as well as the talks at the same event by Chris Tarnovsky on the evolution of chip defense technology [1865]. Finally, for a detailed description of a non-trivial attack, see Chris Gerlinsky's 2016 talk on how he broke VideoCipher [759].