PART I

In the first section of the book, I cover the basics. The first chapter sets out to clarify concepts and terminology by describing the secure distributed systems commonly found in four environments: a bank, an air force base, a hospital, and the home. The second chapter then plunges into the thick of things by describing the threat actors and how they operate. We look at state actors such as the US, Chinese and Russian intelligence communities, about which we now know quite a lot thanks to disclosures by Ed Snowden and others; we describe the cybercrime ecosystem, which we've been studying for some years now; and we also describe non-financial abuses from cyber-bullying and intimate partner abuse up to election manipulation and political radicalisation. This teaches that a wide range of attackers use similar techniques, not just at the technical level but increasingly to deceive and manipulate people.

In the third chapter we therefore turn to psychology. Phishing is a key technique for both online crime and national intelligence gathering; usability failures are exploited all the time, and are really important for safety as well as security. One of the most fruitful areas of security research in recent years has therefore been psychology. Security engineers need to understand how people can be deceived, so we can design systems that make deception harder. We also need to understand how risk perceptions and realities have drifted ever further apart.

The following chapters dig deeper into the technical meat. The fourth chapter is on security protocols, which specify how the players in a system – whether people, computers, phones or other electronic devices – establish and maintain trust. The fifth is on the 'duct tape' that underlies most of the protocols and holds distributed systems together: cryptography. This is the art (and science) of codes and ciphers; but it is much more than a clever means for keeping messages secret from an eavesdropper. Nowadays its job is taking trust from where it exists to where it's needed, maintaining the integrity of security contexts, and much more besides.

The sixth chapter is on access control: how can we keep apart the different apps on a phone, or the different virtual machines or containers on a server, and how can we control the data flows we want to permit between them. Sometimes this can be done cleanly, but often it's hard; web browsers deal with JavaScript code from multiple untrustworthy websites, while home assistants have to deal with multiple people.

The next chapter is on distributed systems. Systems that run on multiple devices have to deal with coordination problems such as concurrency control, fault tolerance, and naming. These take on subtle new meanings when systems must be made resilient against malice as well as against accidental failure. Many systems perform poorly or even fail because their designers don't think through these issues.

The final chapter in this part is on economics. Security economics has grown hugely since this book first appeared in 2001 and helped to launch it as a subject. We now know that many security failures are due to perverse incentives rather than to deficient technical protection mechanisms. (Indeed, the former often explain the latter.) The dependability of a system is increasingly an emergent property that depends on the self-interested striving of large numbers of players; in effect it's an equilibrium in a market. Security mechanisms are not just used to keep 'bad' people out of 'good' systems, but to enable one principal to exert power over another; they are often abused to capture or distort markets. If we want to understand such plays, or to design systems that resist strategic manipulation, we need some game theory and auction theory.

These chapters cover basic material, and largely follow what we teach first-year and second-year undergraduates at Cambridge. But I hope that even experts will find the case studies of interest and value.

CHAPTER

1

What Is Security Engineering?

Out of the crooked timber of humanity, no straight thing was ever made.

– IMMANUEL KANT

The world is never going to be perfect, either on- or offline; so let's not set impossibly high standards for online.

– ESTHER DYSON

1.1 Introduction

Security engineering is about building systems to remain dependable in the face of malice, error, or mischance. As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves.

Security engineering requires cross-disciplinary expertise, ranging from cryptography and computer security through hardware tamper-resistance to a knowledge of economics, applied psychology, organisations and the law. System engineering skills, from business process analysis through software engineering to evaluation and testing, are also important; but they are not sufficient, as they deal only with error and mischance rather than malice. The security engineer also needs some skill at adversarial thinking, just like a chess player; you need to have studied lots of attacks that worked in the past, from their openings through their development to the outcomes.

Many systems have critical assurance requirements. Their failure may endanger human life and the environment (as with nuclear safety and control systems), do serious damage to major economic infrastructure (cash machines and online payment systems), endanger personal privacy (medical record systems), undermine the viability of whole business sectors (prepayment utility meters), and facilitate crime (burglar and car alarms). Security and safety are becoming ever more intertwined as we get software in everything.

Even the perception that a system is more vulnerable or less reliable than it really is can have real social costs.

The conventional view is that while software engineering is about ensuring that certain things happen ("John can read this file"), security is about ensuring that they don't ("The Chinese government can't read this file"). Reality is much more complex. Security requirements differ greatly from one system to another. You typically need some combination of user authentication, transaction integrity and accountability, fault-tolerance, message secrecy, and covertness. But many systems fail because their designers protect the wrong things, or protect the right things but in the wrong way.

Getting protection right thus depends on several different types of process. You have to figure out what needs protecting, and how to do it. You also need to ensure that the people who will guard the system and maintain it are properly motivated. In the next section, I'll set out a framework for thinking about this. Then, in order to illustrate the range of different things that security and safety systems have to do, I will take a quick look at four application areas: a bank, a military base, a hospital, and the home. Once we've given concrete examples of the stuff that security engineers have to understand and build, we will be in a position to attempt some definitions.

1.2 A framework

To build really dependable systems, you need four things to come together. There's policy: what you're supposed to achieve. There's mechanism: the ciphers, access controls, hardware tamper-resistance and other machinery that you use to implement the policy. There's assurance: the amount of reliance you can place on each particular mechanism, and how well they work together. Finally, there's incentive: the motive that the people guarding and maintaining the system have to do their job properly, and also the motive that the attackers have to try to defeat your policy. All of these interact (see Figure 1.1).

As an example, let's think of the 9/11 terrorist attacks. The hijackers' success in getting knives through airport security was not a mechanism failure but a policy one; the screeners did their job of keeping out guns and explosives, but at that time, knives with blades up to three inches were permitted. Policy changed quickly: first to prohibit all knives, then most weapons (baseball bats are now forbidden but whiskey bottles are OK); it's flip-flopped on many details (butane lighters forbidden then allowed again). Mechanism is weak, because of things like composite knives and explosives that don't contain nitrogen. Assurance is always poor; many tons of harmless passengers' possessions are consigned to the trash each month, while less than half of all the real weapons taken through screening (whether accidentally or for test purposes) are spotted and confiscated.

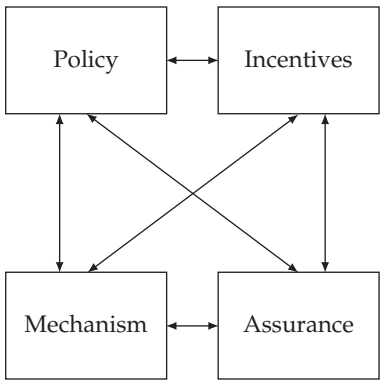

Figure 1.1: – Security Engineering Analysis Framework

Most governments have prioritised visible measures over effective ones. For example, the TSA has spent billions on passenger screening, which is fairly ineffective, while the $100m spent on reinforcing cockpit doors removed most of the risk [1526]. The President of the Airline Pilots Security Alliance noted that most ground staff aren't screened, and almost no care is taken to guard aircraft parked on the ground overnight. As most airliners don't have door locks, there's not much to stop a bad guy wheeling steps up to a plane and placing a bomb on board; if he had piloting skills and a bit of chutzpah, he could file a flight plan and make off with it [1204]. Yet screening staff and guarding planes are just not a priority.

Why are such policy choices made? Quite simply, the incentives on the decision makers favour visible controls over effective ones. The result is what Bruce Schneier calls 'security theatre' – measures designed to produce a feeling of security rather than the reality. Most players also have an incentive to exaggerate the threat from terrorism: politicians to 'scare up the vote' (as President Obama put it), journalists to sell more papers, companies to sell more equipment, government officials to build their empires, and security academics to get grants. The upshot is that most of the damage done by terrorists to democratic countries comes from the overreaction. Fortunately, electorates figure this out over time, and now – nineteen years after 9/11 – less money is wasted. Of course, we now know that much more of our society's resilience budget should have been spent on preparing for pandemic disease. It was at the top of Britain's risk register, but terrorism was politically more sexy. The countries that managed their priorities more rationally got much better outcomes.

Security engineers need to understand all this; we need to be able to put risks and threats in context, make realistic assessments of what might go wrong, and give our clients good advice. That depends on a wide understanding of what has gone wrong over time with various systems; what sort of attacks have worked, what their consequences were, and how they were stopped (if it was

worthwhile to do so). History also matters because it leads to complexity, and complexity causes many failures. Knowing the history of modern information security enables us to understand its complexity, and navigate it better.

So this book is full of case histories. To set the scene, I'll give a few brief examples here of interesting security systems and what they're designed to prevent.

1.3 Example 1 – a bank

Banks operate a lot of security-critical computer systems.

1. A bank's operations rest on a core bookkeeping system. This keeps customer account master files plus a number of journals that record incoming and outgoing transactions. The main threat here is the bank's own staff; about one percent of bank branch staff are fired each year, mostly for petty dishonesty (the average theft is only a few thousand dollars). The traditional defence comes from bookkeeping procedures that have evolved over centuries. For example, each debit against one account must be matched by a credit against another; so money can only be moved within a bank, never created or destroyed. In addition, large transfers typically need two people to authorize them. There are also alarms that look for unusual volumes or patterns of transactions, and staff are required to take regular vacations with no access to the bank's systems.
2. One public face is the bank's automatic teller machines. Authenticating transactions based on a customer's card and personal identification number – so as to defend against both outside and inside attack – is harder than it looks! There have been many epidemics of 'phantom withdrawals' in various countries when local villains (or bank staff) have found and exploited loopholes in the system. Automatic teller machines are also interesting as they were the first large-scale commercial use of cryptography, and they helped establish a number of crypto standards. The mechanisms developed for ATMs have been extended to point-of-sale terminals in shops, where card payments have largely displaced cash; and they've been adapted for other applications such as prepayment utility meters.
3. Another public face is the bank's website and mobile phone app. Most customers now do their routine business, such as bill payments and transfers between savings and checking accounts, online rather than at a branch. Bank websites have come under heavy attack since 2005 from *phishing* – where customers are invited to enter their passwords

at bogus websites. The standard security mechanisms designed in the 1990s turned out to be less effective once criminals started attacking the customers rather than the bank, so many banks now send you a text message with an authentication code. The crooks' reaction is to go to a phone shop, pretend to be you, and buy a new phone that takes over your phone number. This arms race poses many fascinating security engineering problems mixing elements from authentication, usability, psychology, operations and economics.

4. Behind the scenes are high-value messaging systems, used to move large sums between banks; to trade in securities; to issue letters of credit and guarantees; and so on. An attack on such a system is the dream of the high-tech criminal – and we hear that the government of North Korea has stolen many millions by attacks on banks. The defence is a mixture of bookkeeping controls, access controls, and cryptography.
5. The bank's branches may seem large, solid and prosperous, reassuring customers that their money is safe. But the stone facade is theatre rather than reality. If you walk in with a gun, the tellers will give you all the cash you can see; and if you break in at night, you can cut into the safe in minutes with an abrasive wheel. The effective controls center on alarm systems, which are connected to a security company's control center, whose staff check things out by video and call the police if they have to. Cryptography is used to prevent a robber manipulating the communications and making the alarm appear to say 'all's well' when it isn't.

I'll look at these applications in later chapters. Banking computer security is important: until the early 2000s, banks were the main civilian market for many computer security products, so they had a huge influence on security standards.

1.4 Example 2 – a military base

Military systems were the other technology driver back in the 20th century, as they motivated much of the academic research that governments funded into computer security from the early 1980s onwards. As with banking, there's not one application but many.

1. Military communications drove the development of cryptography, going right back to ancient Egypt and Mesopotamia. But it is often not enough to just encipher messages: an enemy who sees traffic encrypted with somebody else's keys may simply locate and attack the transmitter. *Low-probability-of-intercept* (LPI) radio links are one answer; they use tricks that are now adopted in everyday communications such as Bluetooth.

2. Starting in the 1940s, governments spent a lot of money on electronic warfare systems. The arms race of trying to jam enemy radars while preventing the enemy from jamming yours has led to many sophisticated deception tricks, countermeasures, and counter-countermeasures – with a depth, subtlety and range of strategies that are still not found elsewhere. Spoofing and service-denial attacks were a reality there long before blackmailers started targeting the websites of bankers, bookmakers and gamers.
3. Military organisations need to hold some information close, such as intelligence sources and plans for future operations. These are typically labeled 'Top Secret' and handled on separate systems; they may be further restricted in compartments, so that the most sensitive information is known to only a handful of people. For years, attempts were made to enforce information flow rules, so you could copy a file from a *Secret* stores system to a *Top Secret* command system, but not vice versa. Managing multiple systems with information flow restrictions is a hard problem, and the billions that were spent on attempting to automate military security helped develop the access-control technology you now have in your mobile phone and laptop.
4. The problems of protecting nuclear weapons led to the invention of a lot of cool security technology, ranging from provably-secure authentication systems, through optical-fibre alarm sensors, to methods of identifying people using biometrics – including the iris patterns now used to identify all citizens of India.

The security engineer can still learn a lot from this. For example, the military was until recently one of the few customers for software systems that had to be maintained for decades. Now that software and Internet connectivity are finding their way into safety-critical consumer goods such as cars, software sustainability is becoming a much wider concern. In 2019, the European Union passed a law demanding that if you sell goods with digital components, you must maintain those components for two years, or for longer if that's a reasonable expectation of the customer – which will mean ten years for cars and white goods. If you're writing software for a car or fridge that will be on sale for seven years, you'll have to maintain it for almost twenty years. What tools should you use?

1.5 Example 3 – a hospital

From bankers and soldiers we move on to healthcare. Hospitals have a number of interesting protection requirements – mostly to do with patient safety and privacy.

1. Safety usability is important for medical equipment, and is by no means a solved problem. Safety usability failures are estimated to kill about as many people as road traffic accidents – a few tens of thousands a year in the USA, for example, and a few thousand in the UK. The biggest single problem is with the infusion pumps used to drip-feed patients with drugs; a typical hospital might have half-a-dozen makes, all with somewhat different controls, making fatal errors more likely. Safety usability interacts with security: unsafe devices that are also found to be hackable are much more likely to have product recalls ordered as regulators know that the public's appetite for risk is lower when hostile action becomes a possibility. So as more and more medical devices acquire not just software but radio communications, security sensitivities may lead to better safety.
2. Patient record systems should not let all the staff see every patient's record, or privacy violations can be expected. In fact, since the second edition of this book, the European Court has ruled that patients have a right to restrict their personal health information to the clinical staff involved in their care. That means that systems have to implement rules such as "nurses can see the records of any patient who has been cared for in their department at any time during the previous 90 days". This can be harder than it looks. (The US HIPAA legislation sets easier standards for compliance but is still a driver of information security investment.)
3. Patient records are often anonymized for use in research, but this is hard to do well. Simply encrypting patient names is not enough: an enquiry such as "show me all males born in 1953 who were treated for atrial fibrillation on October 19th 2003" should be enough to target former Prime Minister Tony Blair, who was rushed to hospital that day to be treated for an irregular heartbeat. Figuring out what data can be anonymized effectively is hard, and it's also a moving target as we get more and more social and contextual data – not to mention the genetic data of relatives near and far.
4. New technology can introduce poorly-understood risks. Hospital administrators understand the need for backup procedures to deal with outages of power; hospitals are supposed to be able to deal with casualties even if their mains electricity and water supplies fail. But after several hospitals in Britain had machines infected by the Wannacry malware in May 2017, they closed down their networks to limit further infection, and then found that they had to close their accident and emergency departments – as X-rays no longer travel from the X-ray machine to the operating theatre in an envelope, but via a server in a distant town. So a network failure can stop doctors operating when a power failure would not. There were standby generators, but no standby

network. Cloud services can make things more reliable on average, but the failures can be bigger, more complex, and correlated. An issue surfaced by the coronavirus pandemic is accessory control: some medical devices authenticate their spare parts, just as printers authenticate ink cartridges. Although the vendors claim this is for safety, it's actually so they can charge more money for spares. But it introduces fragility: when the supply chain gets interrupted, things are a lot harder to fix.

We'll look at medical system security (and safety too) in more detail later. This is a younger field than banking IT or military systems, but as healthcare accounts for a larger proportion of GNP than either of them in all developed countries, its importance is growing. It's also consistently the largest source of privacy breaches in countries with mandatory reporting.

1.6 Example 4 – the home

You might not think that the typical family operates any secure systems. But just stop and think.

1. You probably use some of the systems I've already described. You may use a web-based electronic banking system to pay bills, and you may have online access to your doctor's surgery so you can order repeat prescriptions. If you're diabetic then your insulin pump may communicate with a docking station at your bedside. Your home burglar alarm may send an encrypted 'all's well' signal to the security company every few minutes, rather than waking up the neighborhood when something happens.
2. Your car probably has an electronic immobilizer. If it was made before about 2015, the car unlocks when you press a button on the key, which sends an encrypted unlock command. If it's a more recent model, where you don't have to press any buttons but just have the key in your pocket, the car sends an encrypted challenge to the key and waits for the right response. But eliminating the button press meant that if you leave your key near the front door, a thief might use a radio relay to steal your car. Car thefts have shot up since this technology was introduced.
3. Your mobile phone authenticates itself to the network by a cryptographic challenge-response protocol similar to the ones used in car door locks and immobilizers, but the police can use a false base station (known in Europe as an IMSI-catcher, and in America as a Stingray) to listen in. And, as I mentioned above, many phone companies are relaxed about selling new SIM cards to people who claim their phones have been stolen; so a crook might steal your phone number and use this to raid your bank account.

4. In over 100 countries, households can get prepayment meters for electricity and gas, which they top up using a 20-digit code that they buy from an ATM or an online service. It even works off-grid; in Kenyan villages, people who can't afford $200 to buy a solar panel can get one for $2 a week and unlock the electricity it generates using codes they buy with their mobile phones.
5. Above all, the home provides a haven of physical security and seclusion. This is changing in a number of ways. Burglars aren't worried by locks as much as by occupants, so alarms and monitoring systems can help; but monitoring is also becoming pervasive, with many households buying systems like Alexa and Google Home that listen to what people say. All sorts of other gadgets now have microphones and cameras as voice and gesture interfaces become common, and the speech processing is typically done in the cloud to save battery life. By 2015, President Obama's council of advisers on science and technology was predicting that pretty soon every inhabited space on earth would have microphones that were connected to a small number of cloud service providers. (The USA and Europe have quite different views on how privacy law should deal with this.) One way or another, the security of your home may come to depend on remote systems over which you have little control.

Over the next few years, the number of such systems is going to increase rapidly. On past experience, many of them will be badly designed. For example, in 2019, Europe banned a children's watch that used unencrypted communications to the vendor's cloud service; a wiretapper could download any child's location history and cause their watch to phone any number in the world. When this was discovered, the EU ordered the immediate safety recall of all watches [903].

This book aims to help you avoid such outcomes. To design systems that are safe and secure, an engineer needs to know about what systems there are, how they work, and – at least as important – how they have failed in the past. Civil engineers learn far more from the one bridge that falls down than from the hundred that stay up; exactly the same holds in security engineering.

1.7 Definitions

Many of the terms used in security engineering are straightforward, but some are misleading or even controversial. There are more detailed definitions of technical terms in the relevant chapters, which you can find using the index. In this section, I'll try to point out where the main problems lie.

The first thing we need to clarify is what we mean by *system*. In practice, this can denote:

1. a product or component, such as a cryptographic protocol, a smartcard, or the hardware of a phone, a laptop or server;
2. one or more of the above plus an operating system, communications and other infrastructure;
3. the above plus one or more applications (banking app, health app, media player, browser, accounts/payroll package, and so on – including both client and cloud components);
4. any or all of the above plus IT staff;
5. any or all of the above plus internal users and management;
6. any or all of the above plus customers and other external users.

Confusion between the above definitions is a fertile source of errors and vulnerabilities. Broadly speaking, the vendor and evaluator communities focus on the first and (occasionally) the second of them, while a business will focus on the sixth (and occasionally the fifth). We will come across many examples of systems that were advertised or even certified as secure because the hardware was, but that broke badly when a particular application was run, or when the equipment was used in a way the designers didn't anticipate. Ignoring the human components, and thus neglecting usability issues, is one of the largest causes of security failure. So we will generally use definition 6; when we take a more restrictive view, it should be clear from the context.

The next set of problems comes from lack of clarity about who the players are and what they're trying to prove. In the literature on security and cryptology, it's a convention that principals in security protocols are identified by names chosen with (usually) successive initial letters – much like hurricanes, except that we use alternating genders. So we see lots of statements such as "Alice authenticates herself to Bob". This makes things much more readable, but can come at the expense of precision. Do we mean that Alice proves to Bob that her name actually is Alice, or that she proves she's got a particular credential? Do we mean that the authentication is done by Alice the human being, or by a smartcard or software tool acting as Alice's agent? In that case, are we sure it's Alice, and not perhaps Carol to whom Alice lent her card, or David who stole her phone, or Eve who hacked her laptop?

By a *subject* I will mean a physical person in any role including that of an operator, principal or victim. By a *person*, I will mean either a physical person or a legal person such as a company or government[1].

[1]The law around companies may come in handy when we start having to develop rules around AI. A company, like a robot, may be immortal and have some functional intelligence – but without consciousness. You can't jail a company but you can fine it.

A *principal* is an entity that participates in a security system. This entity can be a subject, a person, a role, or a piece of equipment such as a laptop, phone, smartcard, or card reader. A principal can also be a communications channel (which might be a port number, or a crypto key, depending on the circumstance). A principal can also be a compound of other principals; examples are a group (Alice or Bob), a conjunction (Alice and Bob acting together), a compound role (Alice acting as Bob's manager) and a delegation (Bob acting for Alice in her absence).

Beware that groups and roles are not the same. By a *group* I will mean a set of principals, while a *role* is a set of functions assumed by different persons in succession (such as 'the officer of the watch on the USS Nimitz' or 'the president for the time being of the Icelandic Medical Association'). A principal may be considered at more than one level of abstraction: e.g. 'Bob acting for Alice in her absence' might mean 'Bob's smartcard representing Bob who is acting for Alice in her absence' or even 'Bob operating Alice's smartcard in her absence'. When we have to consider more detail, I'll be more specific.

The meaning of the word *identity* is controversial. When we have to be careful, I will use it to mean a correspondence between the names of two principals signifying that they refer to the same person or equipment. For example, it may be important to know that the Bob in 'Alice acting as Bob's manager' is the same as the Bob in 'Bob acting as Charlie's manager' and in 'Bob as branch manager signing a bank draft jointly with David'. Often, identity is abused to mean simply 'name', an abuse entrenched by such phrases as 'user identity' and 'citizen identity card'.

The definitions of *trust* and *trustworthy* are often confused. The following example illustrates the difference: if an NSA employee is observed in a toilet stall at Baltimore Washington International airport selling key material to a Chinese diplomat, then (assuming his operation was not authorized) we can describe him as 'trusted but not trustworthy'. I use the NSA definition that a *trusted* system or component is one whose failure can break the security policy, while a *trustworthy* system or component is one that won't fail.

There are many alternative definitions of trust. In the corporate world, trusted system might be 'a system which won't get me fired if it gets hacked on my watch' or even 'a system which we can insure'. But when I mean an approved system, an insurable system or an insured system, I'll say so.

The definition of *confidentiality* versus *privacy* versus *secrecy* opens another can of worms. These terms overlap, but are not exactly the same. If my neighbor cuts down some ivy at our common fence with the result that his kids can look into my garden and tease my dogs, it's not my confidentiality that has been invaded. And the duty to keep quiet about the affairs of a former employer is a duty of confidence, not of privacy.

The way I'll use these words is as follows.

- *Secrecy* is an engineering term that refers to the effect of the mechanisms used to limit the number of principals who can access information, such as cryptography or computer access controls.
- *Confidentiality* involves an obligation to protect some other person's or organisation's secrets if you know them.
- *Privacy* is the ability and/or right to protect your personal information and extends to the ability and/or right to prevent invasions of your personal space (the exact definition of which varies from one country to another). Privacy can extend to families but not to legal persons such as corporations.

For example, hospital patients have a right to privacy, and in order to uphold this right the doctors, nurses and other staff have a duty of confidence towards their patients. The hospital has no right of privacy in respect of its business dealings but those employees who are privy to them may have a duty of confidence (unless they invoke a whistleblowing right to expose wrongdoing). Typically, privacy is secrecy for the benefit of the individual while confidentiality is secrecy for the benefit of the organisation.

There is a further complexity in that it's often not sufficient to protect data, such as the contents of messages; we also have to protect metadata, such as logs of who spoke to whom. For example, many countries have laws making the treatment of sexually transmitted diseases secret, and yet if a private eye could observe you exchanging encrypted messages with a sexually-transmitted disease clinic, he might infer that you were being treated there. In fact, a key privacy case in the UK turned on such a fact: a model in Britain won a privacy lawsuit against a tabloid newspaper which printed a photograph of her leaving a meeting of Narcotics Anonymous. So *anonymity* can be just as important a factor in privacy (or confidentiality) as secrecy. But anonymity is hard. It's difficult to be anonymous on your own; you usually need a crowd to hide in. Also, our legal codes are not designed to support anonymity: it's much easier for the police to get itemized billing information from the phone company, which tells them who called whom, than it is to get an actual wiretap. (And it's often more useful.)

The meanings of *authenticity* and *integrity* can also vary subtly. In the academic literature on security protocols, authenticity means integrity plus freshness: you have established that you are speaking to a genuine principal, not a replay of previous messages. We have a similar idea in banking protocols. If local banking laws state that checks are no longer valid after six months, a seven month old uncashed check has integrity (assuming it's not been altered) but is no longer valid. However, there are some strange edge cases. For example, a police crime scene officer will preserve the integrity of a forged check – by

placing it in an evidence bag. (The meaning of integrity has changed in the new context to include not just the signature but any fingerprints.)

The things we don't want are often described as hacking. I'll follow Bruce Schneier and define a *hack* as something a system's rules permit, but which was unanticipated and unwanted by its designers [1682]. For example, tax attorneys study the tax code to find loopholes which they develop into tax avoidance strategies; in exactly the same way, black hats study software code to find loopholes which they develop into exploits. Hacks can target not just the tax system and computer systems, but the market economy, our systems for electing leaders and even our cognitive systems. They can happen at multiple layers: lawyers can hack the tax code, or move up the stack and hack the legislature, or even the media. In the same way, you might try to hack a cryptosystem by finding a mathematical weakness in the encryption algorithm, or you can go down a level and measure the power drawn by a device that implements it in order to work out the key, or up a level and deceive the device's custodian into using it when they shouldn't. This book contains many examples. In the broader context, hacking is sometimes a source of significant innovation. If a hack becomes popular, the rules may be changed to stop it; but it may also become normalised (examples range from libraries through the filibuster to search engines and social media).

The last matter I'll clarify here is the terminology that describes what we're trying to achieve. A *vulnerability* is a property of a system or its environment which, in conjunction with an internal or external *threat*, can lead to a *security failure*, which is a breach of the system's security policy. By *security policy* I will mean a succinct statement of a system's protection strategy (for example, "in each transaction, sums of credits and debits are equal, and all transactions over $1,000,000 must be authorized by two managers"). A *security target* is a more detailed specification which sets out the means by which a security policy will be implemented in a particular product – encryption and digital signature mechanisms, access controls, audit logs and so on – and which will be used as the yardstick to evaluate whether the engineers have done a proper job. Between these two levels you may find a *protection profile* which is like a security target, except written in a sufficiently device-independent way to allow comparative evaluations among different products and different versions of the same product. I'll elaborate on security policies, security targets and protection profiles in Part 3. In general, the word *protection* will mean a property such as confidentiality or integrity, defined in a sufficiently abstract way for us to reason about it in the context of general systems rather than specific implementations.

This somewhat mirrors the terminology we use for safety-critical systems, and as we are going to have to engineer security and safety together in ever more applications it is useful to keep thinking of the two side by side.

In the safety world, a *critical* system or component is one whose failure could lead to an accident, given a *hazard* – a set of internal conditions or external circumstances. *Danger* is the probability that a hazard will lead to an accident, and *risk* is the overall probability of an accident. Risk is thus hazard level combined with danger and *latency* – the hazard exposure and duration. *Uncertainty* is where the risk is not quantifiable, while *safety* is freedom from accidents. We then have a *safety policy* which gives us a succinct statement of how risks will be kept below an acceptable threshold (and this might range from succinct, such as "don't put explosives and detonators in the same truck", to the much more complex policies used in medicine and aviation); at the next level down, we might find a *safety case* having to be made for a particular component such as an aircraft, an aircraft engine or even the control software for an aircraft engine.

1.8 Summary

'Security' is a terribly overloaded word, which often means quite incompatible things to different people. To a corporation, it might mean the ability to monitor all employees' email and web browsing; to the employees, it might mean being able to use email and the web without being monitored.

As time goes on, and security mechanisms are used more and more by the people who control a system's design to gain some commercial advantage over the other people who use it, we can expect conflicts, confusion and the deceptive use of language to increase.

One is reminded of a passage from Lewis Carroll:

> *"When I use a word," Humpty Dumpty said, in a rather scornful tone, "it means just what I choose it to mean – neither more nor less." "The question is," said Alice, "whether you can make words mean so many different things." "The question is," said Humpty Dumpty, "which is to be master – that's all."*

The security engineer must be sensitive to the different nuances of meaning that words acquire in different applications, and be able to formalize what the security policy and target actually are. That may sometimes be inconvenient for clients who wish to get away with something, but, in general, robust security design requires that the protection goals are made explicit.