CHAPTER

3

Passwords

Humans are incapable of securely storing high-quality cryptographic keys, and they have unacceptable speed and accuracy when performing cryptographic operations. (They are also large, expensive to maintain, difficult to manage, and they pollute the environment. It is astonishing that these devices continue to be manufactured and deployed. But they are sufficiently pervasive that we must design our protocols around their limitations.)

—KAUFMAN, PERLMAN, AND SPECINER [444]

Taking care of old-fashioned access control tokens such as metal keys is a matter of common sense. But common sense is not always adequate for the measures used to protect computer systems. The human-machine gap causes security problems in a number of contexts, from straightforward system administration to the ways in which users mismanage security products such as encryption software [803]. (I won't use the fashionable euphemism "human computer interface": "chasm" might be better.) However, most of the problems arise in a simple context in which they are relatively easy to analyze and discuss—the management of passwords.

In addition to things that are "obviously" passwords, such as the password you use to log on to your computer and the PIN that activates your bank card, there are many other things (and combinations of things) that have an equivalent effect. The most notorious are the likes of Social Security numbers and your mother's maiden name, which many organizations use to recognize you. For example, AT&T's wireless service contract states that anyone who knows your name, address, phone number and the last four digits of your social security number is authorized to make changes to your account; it also disclaims all liability for lack of privacy [201].

The ease with which such data can be guessed or found out from more or less public sources has given rise to a huge *identity theft* industry [285]. Criminals obtain credit cards, mobile phones, and other assets in your name, loot them, and leave you to sort out the mess. In the United States, about half a million people are the victims of this kind of fraud each year.

Passwords are one of the biggest practical problems facing security engineers today. They are the (often shaky) foundation on which much of information security is built. Remembering a password is contingent on frequent use (so that passwords are imprinted well on memory) and consistent context (so that different passwords do not

interfere with each other in memory). Neither of these conditions is met when people are asked to choose passwords for a large number of Web sites that they visit rarely. So as they become principals in more and more electronic systems, the same passwords get used over and over again. Not only may attacks be carried out by outsiders guessing passwords, but by insiders in other systems.

3.1 Basics

In a typical system, human users must authenticate themselves to a client (which may be a PC, a mobile phone, an ATM, or whatever), and the client in turn authenticates itself to one or more servers or services (such as an electronic banking system or a phone company). As explained in Chapter 2, "Protocols," authenticating electronic devices to each other is a more or less manageable problem (at least in theory). Authenticating people to devices is more difficult.

There are basically three ways to do it. The first is that the person retains physical control of the device—as with a remote car-door key, a PDA, or even a laptop computer. The second is that he or she presents something he or she knows, such as a password. The third is to use a biometric, such as a fingerprint or iris pattern. (These options are commonly summed up as "something you have, something you know, or something you are.") For reasons of cost, most systems take the second option. Even where we use a physical token such as a hand-held password generator, it is common to use a password as well to lock it.

So passwords matter, and managing them is a serious real-world problem. We'll look at the human issues first, then at the different kinds of attack model, and finally at technical attacks and defenses. All of these issues are important, so tackling only one of them is likely to lead to a bad design.

3.2 Applied Psychology Issues

There are basically three types of concern:

- Will the user break the system security by disclosing the password to a third party, whether accidentally, on purpose, or as a result of deception?
- Will the user enter the password correctly with a high enough probability?
- Will users remember the password, or will they have to either write it down or choose one that's easy for the attacker to guess?

3.2.1 Social Engineering

One of the most severe practical threats to the confidentiality of information is that the attacker will extract it directly, from people who are authorized to access it, by telling some plausible untruth. This attack, known as *social engineering*, will be discussed at greater length in Chapter 8, which deals with medical systems, as it is the main current threat to medical privacy. The typical perpetrator is an insurance investigator who phones a hospital or doctor's office pretending to be a doctor involved in the emergency care of the target of investigation. This technique, also known as 'blagging' in

Britain and 'pretexting' in America, is widely used to extract information from banks, insurance companies, and other firms that hold personal information; some people earn a living at it [261].

Passwords are often extracted by false pretext phone calls. A harrassed system administrator is called once or twice on trivial matters by someone who claims to be a very senior manager's personal assistant; once he has accepted the caller's story, she calls and urgently demands a high-level password on some plausible pretext. Unless an organization has well-thought-out policies, attacks of this kind are very likely to work. In a systematic experimental study, for example, 336 computer science students at the University of Sydney were sent an email message asking them to supply their password on the pretext that it was required to "validate" the password database after a suspected break-in. 138 of them returned a valid password. Some were suspicious: 30 returned a plausible-looking but invalid password, while over 200 changed their passwords without official prompting. But very few of them reported the email to authority [354].

One company controls this vulnerability with a policy that states: "The root password for each machine shall be too long to remember, at least 16 alpha and numeric characters chosen at random by the system; it shall be written on a piece of paper and kept in an envelope in the room where the machine is located; it may never be divulged over the telephone or used over the network; it may only be entered at the console of the machine that it controls." If a rule like this is rigidly enforced throughout an organization, a pretext attack on a root password becomes conspicuous, and is much less likely to succeed.

Another approach, used at the NSA, is to have different-colored internal and external telephones that are not connected to each other, and rules that when the external phone in a room is off-hook, classified material can't even be discussed in the room, let alone on the phone. A somewhat less extreme approach (used at our laboratory) is to have different ring tones for internal and external calls. This works as long as you have alert system administrators. Physical authentication devices, like the password generator discussed in Chapter 2, are even better but are often too expensive, incompatible with legacy systems, or contrary to some policy (whether reasonable or not).

3.2.2 Difficulties with Reliable Password Entry

The second human issue is that if a password is too long or complex, the user might have difficulty entering it correctly. A long random password may confuse the person entering it, and if the operation they are trying to perform is urgent, this might have safety or other implications.

One application in which this is important is encrypted access codes. By quoting a reservation number, we get access to a hotel room or rental car. Airline ticketing is going this way, with many operators giving passengers a number to quote at the departure gate rather than a boarding card. As the numbers get longer, what happens to the error rate?

An interesting study was done in South Africa, in the context of the prepaid electricity meters used to sell electricity in areas where the customers have no credit rating and often not even an address. With one make of meter, the customer hands some money to a sales agent, and in return gets one or more 20-digit numbers printed out on a receipt. He takes this receipt home and enters the numbers at a keypad in his meter. These numbers are encrypted commands, whether to dispense electricity, to change the tariff or whatever; the meter decrypts them and acts on them.

When this meter was introduced, there was concern that since about a third of the population was illiterate, and people might get lost halfway through entering the number, this meter might be unusable in practice. But it turned out that illiteracy was not a problem; even people who could not read had no difficulty with numbers ("Everybody can use a phone," as one of the engineers said). Entry errors were a greater problem, but were solved by printing the 20 digits in two rows, containing, respectively, three and two groups of four digits [39].

A quite different application is the firing codes for U.S. nuclear weapons. These consist of only 12 decimal digits. If they are ever used, it is likely that the operators will be under the most extreme stress, and possibly using improvised or obsolete communications channels. Experiments suggested that 12 digits was the maximum that could be conveyed reliably in such circumstances.

3.2.3 Difficulties with Remembering the Password

The greatest source of complaints about passwords is the fact that most people find them hard to remember [146, 823]. Twelve to twenty digits may be fine when they can be simply copied from a telegram or a meter ticket, but when customers are expected to memorize passwords, they either choose values that are easy for attackers to guess, or write them down, or both.

The problems are not limited to computer access. For example, one chain of hotels in France introduced completely unattended service. You would turn up at the hotel, swipe your credit card in the reception machine, and get a receipt with a numerical access code that would unlock your room door. To keep costs down, the rooms did not have en suite bathrooms, so guests had to use communal facilities. The usual failure mode was that a guest, having gone to the bathroom, would forget his access code. Unless he had taken the receipt with him, he'd end up having to sleep on the bathroom floor until the staff arrived the following morning.

Problems related to password memorability can be discussed under two main headings: design errors, and operational failures.

3.2.3.1 Design Errors

Attempts to design systems so as to make passwords memorable are a frequent source of severe design errors—especially with the many systems being built rapidly by unskilled people for electronic business. An instructive, and important, example of how not to do it is to ask customers for "your mother's maiden name." Many banks, government departments, and other organizations authenticate their customers in this way. There are two rather obvious problems: first, your mother's maiden name is easy for a thief to find out, whether by asking around, chasing birth and marriage records, or using online genealogical databases. Second, even if you decide that from now on your mother's maiden name is going to be, say, Yngstrom (or even `yGt5r4ad`), rather than Smith, there are problems. You might break your credit card agreement, and perhaps invalidate your insurance cover, by giving false data.

Moreover, asking for a maiden name makes assumptions that don't hold for all cultures (Icelanders have no surnames, and women from many other countries don't change their names on marriage). There might be no provision for changing such a password, so if it ever becomes known to a thief you could have to close and reopen bank accounts. Finally, you will be asked to give it to a lot of organizations, any one of

which might have a crooked employee. You could always tell "Yngstrom" to your bank. "Jones" to the phone company, "Geraghty" to the travel agent, and so on; but data are shared extensively between companies, so you could easily end up confusing their systems (not to mention yourself).

Slightly more thoughtfully designed e-commerce sites ask for a password explicitly rather than a maiden name. But the sheer number of applications for which the average person is asked to use a password nowadays exceeds the powers of human memory. So either customers will write passwords down (despite being told not to) or they will use the same password for many different purposes. Thus, the password you use to authenticate the customer of the electronic banking system you've just designed, is quite possibly known to a Mafia-operated porn site as well.

The risk you face as a consumer is not just a direct loss through identity theft or fraud. Badly designed password mechanisms can undermine your credibility and can cause you to lose a genuine legal claim. For example, if a thief manages to forge a copy of your cash machine card, then loots your bank account, the bank will ask whether you have ever shared your personal identification number with any other person or company. If you admit to using the same number for your mobile phone, the bank may well say that either you were grossly negligent by allowing someone to see you using the phone, or somebody at the phone company must be to blame. In either case, it's up to you to find them and sue them.

Some organizations try to find other security information. My bank asks its business customers the value of the last check from their account that was cleared. In theory, this could be a good system: it has the advantage that even if someone compromises my password—such as by overhearing me doing a transaction on the telephone—the security of the system usually recovers more or less automatically. The implementation details bear some attention though. When this system was first introduced, I wondered whether a supplier, to whom I'd just written a check, had a chance of impersonating me. I concluded that asking for the last three checks' values would be safer. But the problem I actually had was different. Having given the checkbook to our accountant for the annual audit, I couldn't authenticate myself to get a balance over the phone and had to visit the branch.

Attempts to find alternative solutions have more often hit the rocks. One bank sent its customers a letter warning them against writing down their PIN, and instead supplied a distinctive piece of cardboard on which they were supposed to conceal their PIN in the following way: suppose your PIN is 2256. Choose a four-letter word, say `blue`. Write these four letters down in the second, second, fifth, and sixth columns of the card, respectively, as shown in Figure 3.1. Then fill up the empty boxes with random letters.

1	2	3	4	5	6	7	8	9	0
	b								
	l								
				u					
					e				

Figure 3.1 A bad mnemonic system for bank PINs.

This is clearly a bad idea. Even if the random letters aren't written in a slightly different way, a quick check shows that a 4 by 10 matrix of random letters may yield about two dozen words (unless there's an "s" on the bottom row, when you can get 40 to 50). So the odds that the thief can guess the PIN, given three attempts, have just shortened from 1 in 3000-odd to 1 in 8.

Some banks allow customers to choose their own PINs. It is believed that about a third of customers use a birthdate, in which case the odds against the thief are now a bit over 100 to 1 (and much shorter if the thief knows the victim). Even if this risk is thought acceptable, the PIN might still be set to the same value as the PIN used with a mobile phone that's shared with family members. To analyze this problem, we have to consider a number of different threat models, which we'll come to in the next section.

3.2.3.2 Operational Issues

A failure to think through the sort of rules that organizations should make, and enforce, to support the password mechanisms they have implemented has led to some really spectacular cases. One important case in Britain in the late 1980s was *R v. Gold and Schifreen*. The defendants saw a phone number for the development system for Prestel (an early public email service run by British Telecom) in a note stuck on a terminal at an exhibition. They dialed in later, and found that the welcome screen had an all-powerful maintenance password displayed on it. They tried this on the live system, too, and it worked! They proceeded to take over the Duke of Edinburgh's electronic mail account, and sent mail 'from' him to someone they didn't like, announcing the award of a knighthood. This crime so shocked the establishment that when prosecutors failed to convict the defendants under the laws then in force, Parliament passed Britain's first computer crime law.

Placing an administrator password in an envelope taped to the side of a workstation in an office that is always attended or locked may in some circumstances be reasonable practice. The people who can get at it are those who can physically access the machine anyway. But if you operate a policy like this, then you have to see to it that people understand the reasoning behind it, and don't think that the password can as easily be left on a logon screen. For someone to use the same administrator password for the live system as in the development environment is much less excusable.

A similar and very general error is failing to reset the default passwords supplied with certain system services. For example, one top-selling dial access system in the 1980s had a default software support user name of 999999 and a password of 9999. It also had a default supervisor name of 777777, with a password of 7777. Most sites didn't change these passwords, and many of them were hacked once the practice became widely known. Failure to change default passwords as supplied by the equipment vendor has affected many kinds of computer, some cryptographic equipment, and even mobile phones (where many users never bother to change an installed PIN of 0000).

3.3 System Issues

After gaining an understanding the psychology of the users, the next step is to understand that of the attackers. Just as we can only talk about the soundness of a security protocol in the context of a specific threat model, so we can only judge whether a given password scheme is sound by considering the type of attacks we are trying to defend against. Broadly speaking, these are:

Targeted attack on one account. An intruder tries to guess a particular user's password. He might try to guess the PIN for Bill Gates's bank account, or a rival's logon password at the office, in order to do mischief directly.

Attempt to penetrate any account on a system. The intruder tries to get a logon as any user of the system. This might be to steal service directly (e.g., use a phone card service without paying) or as a stepping stone to a wider attack.

Attempt to penetrate any account on any system. The intruder wants an account on any system in a given domain; it doesn't matter which one. For example, common teenage hacker motives are to get a place to hide pirated software or pornography, or a platform from which attacks can be launched anonymously on other systems. More serious threats come from skilled people attacking a target organization. A spy seeking classified information might initially try to hack any computer in the `.mil` namespace, while a private eye tasked to get access to Microsoft's intranet might only need a logon to some random machine in `microsoft.com`.

Service denial attack The attacker may wish to prevent the legitimate user from using the system. This might be targeted on a particular account (such as cancelling somebody's credit cards in order to annoy them) or systemwide.

This taxonomy is useful because it helps us ask many relevant questions when selecting or designing a password system. However, there are other issues that interact with the type of attacks that are expected and the kind of countermeasures that can be used.

3.3.1 Protecting Oneself or Others?

First, to what extent does the system need to protect users from each other? In some systems—such as mobile phone systems and cash machine systems—no one should be able to use the service at someone else's expense. It is assumed that the attackers are already legitimate users of the system. So systems are (or at least should be) carefully designed so that knowledge of one user's password will not allow another identifiable user's account to be compromised: they provide *multilateral security* (which we discuss at greater length in Chapter 8). A user who chooses a password that is easy to guess harms only himself, and so a wide variation in password strength can perhaps be tolerated. (Bear in mind that the passwords people choose are very often easy for their spouses or partners to guess [146], so some thought needs to be given to issues such as what happens when a cheated partner seeks vengeance. This is a common enough problem.)

But with many systems, it can be bad news if even one enemy gains access. Operating systems such as Unix and Windows may have been designed to protect one user against accidental interference by another, but they are not hardened to protect against capable malicious actions by other users. These systems have many well-publicized vulnerabilities, with more being published constantly on the Web. A competent opponent who can get a single account on a shared computer system can usually become the system administrator fairly quickly; and from there he can do whatever he likes. The typical exploitation path is thus *outsider* to *normal user* to *administrator*, with the first of these steps being the hard one. So it may not be a good idea to let users choose whatever password they like. With military systems in particular, it is common to assign users random passwords in order to guarantee a minimum password strength. (I'll have more to say on this later.)

3.3.2 Intrusion Detection Issues

The second question concerns the manner in which the password system might interact with an intrusion detection system. Organizations such as banks often have a rule that a terminal and user account are frozen after three bad password attempts; it is then necessary to contact an administrator to reactivate them. This could be rather dangerous in a military system, as an enemy who got access to the network could use a flood of false logon attempts to mount a service denial attack; if given a list of all the user names on a machine, it might well be possible to take it out of service completely.

It's not just military systems where you have to pay attention to this. Telephone calling cards are another example; they usually have a prefix, followed by the local number to which they're billed, and a four-digit PIN. One phone company scans calling card numbers and cancels any for which more than one PIN appears. This leaves them wide open to anyone who wants to cancel someone else's card. (It also doesn't stop the crook who wants to guess a valid card number, as he can just try the same PIN with a whole lot of different local phone numbers.)

The design of intrusion detection systems varies greatly by what they are supposed to achieve. They can range from simple threshold alarms, which go off after three bad logon attempts to a particular account, up through much more sophisticated and distributed systems designed to cope with intruders who might try one password on many accounts, or on one account on each of many machines, or whatever. There's more on intrusion detection in Chapter 18; here, I'm just flagging up the fact that password and intrusion detection policies interact.

3.3.3 Can Users Be Trained?

The third question is whether users can be trained and disciplined. In a corporate or military environment—and even to some extent in a university—you can control your user population. You can teach them how to choose good passwords; you can give negative feedback if they choose bad ones; you can issue them with random passwords, and order that if these passwords are written down they must be treated the same as the data they protect (so 'Top Secret' passwords must be sealed in an envelope, in a safe, in a room that's locked when not occupied, in a building patrolled by guards). You can see to it that only cleared people have access to the terminals where the passwords can be used. You can send the guards round at night to check that no-one's left a note of a

password lying around. You can operate a *clean desk* policy so that nothing can be overlooked in a pile of papers in plain sight.

Colleagues and I studied the benefits that can be obtained by training users [815]. While writing this book, I could not find any account of experiments on this that would hold water by the standards of applied psychology (i.e., randomized controlled trials with big enough groups for the results to be statistically significant). The closest I found was a study of the recall rates, forgetting rates, and guessing rates of various types of password [146]; this is valuable, but doesn't tell us the actual (as opposed to likely) effects of giving users various kinds of advice. We therefore selected three groups of about a hundred volunteers from our first-year science students.

- The red (control) group was given the usual advice (devise a password of at least six characters long, including one nonletter).
- The green group was told to think of a passphrase and select letters from it to build a password. Thus, "It's 12 noon and I am hungry" would give `I S12&IAH`.
- The yellow group was told to select eight characters (alpha or numeric) at random from a table we gave them, write them down, and destroy the note after a week or two once they'd memorized the password.

What we expected to find was that the red group's passwords would be easier to guess than the green group's, which would in turn be easier than the yellow group's; and that the yellow group would have the most difficulty remembering their passwords (or would be forced to reset them more often), followed by green and then red. But that's not what we found.

About 30 percent of the control group chose passwords that could be guessed using cracking software (which I discuss later), versus about 10 percent for the other two groups. So passphrases and random passwords seemed to be about equally effective. When we looked at password reset rates, there was no significant difference between the three groups. When we asked the students whether they'd found their passwords hard to remember (or had written them down), the yellow group had significantly more problems than the other two; but there was no significant difference between red and green.

The conclusions we drew were as follows.

- For those users who follow instructions, the use of passwords based on mnemonic phrases offers the best of both worlds. They are as easy to remember as naively selected passwords, and as hard to guess as random passwords.
- Merely using mnemonic passwords or random passwords does not help much, as the problem then becomes one of *user compliance*. A significant number of users (perhaps a third of them) just don't do what they're told.

So, while centrally assigned, randomly selected passwords may be a good strategy for the military, its value comes from the fact that the passwords are centrally assigned (thus compelling user compliance) rather than from the fact that they're random (mnemonic phrases would do just as well).

However, there are at least two cases where centrally assigned passwords may be inappropriate. The first is where a user controls access to a resource that the organization should not be able to override. Where digital signatures are used to provide evidence, and a user's digital signing key is protected by a password, then assigning this pass-

word centrally could enable the system administrator to get at the signing key and forge messages, which would destroy the evidential value of the signature.

The second, and more subtle, case is systems that offer a service to the public. Whether you offer a service through dedicated terminals, such as cash machines or mobile phones, or over the Net to standard PCs, you can't expect to train and discipline your users; and if you try to, there is a real risk that a judge will find your contract terms unreasonable.

Perhaps the ideal solution is instruct users to choose mnemonic passwords, and to have a password cracking program installed as a password filter; users who try to choose a password on its guessing list are told to try again. More empirical psychological research on this topic is needed.

3.3.4 The Growing Famine for Security Data

The fourth question is the really hard one: will your users compromise their passwords by using them on other systems?

People who are allowed to select their own password or PIN will often choose the same one for a number of systems, so it's easy to remember. If you don't let customers change their PINs, some of them will write them down. You can forbid this in their contract, but they'll do it anyway. Some people will just take their business elsewhere (given the option, I prefer to use Web sites that don't ask for a password at all, regardless of whether I choose it or they do).

There is a severe shortage of good security information by which people can be identified and can authorize actions. Attempts to solve the problem by issuing people with "multifunction" smartcards have so far foundered on arguments over whose logo will go on front and who will control the mailing list. It would require less expenditure on infrastructure if we could get people to authorize transactions using existing equipment such as mobile phones. But even if, in a few years' time, everyone in the world has a third-generation mobile phone capable of doing banking transactions and of receiving encrypted text messages containing authorization codes for Web-based e-commerce transactions, there will remain a whole host of technical problems. These include the chosen protocol attack, which we discussed in the previous chapter, and the difficulties of preventing programs from interfering with each other, which we will discuss in the next.

Even more serious problems are likely to arise from business and legal issues, such as what if a single company gets control of everyone's credit card data, or purchase history data, or both. We'll discuss these issues in Chapters 19 through 21.

3.4 Technical Protection of Passwords

A broad range of attacks can be used to recover other people's passwords. Some of them target the password entry mechanism, while others exploit the way that passwords are stored.

3.4.1 Attacks on Password Entry

Password entry is often poorly protected.

3.4.1.1 Interface Design

Sometimes the problem is thoughtless interface design. For example, some very common models of cash machine had a vertical keyboard at head height, making it simple for a pickpocket to watch a customer enter her PIN before lifting her purse from her handbag. The keyboards were at a reasonable height for the men who designed them, but women—and men in many countries are a few inches shorter and were highly exposed. Ironically, one of these machines "protected client privacy" by forcing the customer to gaze at the screen through a narrow slot. Your balance was private, but your PIN was not!

Many pay telephones have a similar problem, and *shoulder surfing* of calling card details (as it's known in the industry) has been endemic at some locations such as major U.S. train stations and airports. For that reason, I usually cover my dialling hand with my body or my other hand when entering a card number or PIN in a public place—but systems shouldn't be designed on the assumption that all customers will do this.

3.4.1.2 Eavesdropping

Taking care with password entry may stop the bad guys looking over your shoulder as you use your calling card at an airport telephone, but it won't stop all the eavesdropping attacks. For example, a hotel manager might abuse his switchboard facilities to log the keystrokes you enter at the phone in your room. That way, he might get the credit card number you used to buy a ticket from an automated service; and if this isn't the card number you use to pay your hotel bill, he can plunder your account with much less risk.

Many networked computer systems still send a password in clear over a local area network for checking at a server; anyone who can program a machine on the network, or attach his own sniffer equipment, can harvest them. This is one reason that Microsoft adopted the Kerberos authentication protocol for Windows 2000—the cleartext password is not transmitted over the network. (NT v 4 used a proprietary authentication protocol.)

3.4.1.3 The Need for Trusted Path

The machine to which you log on may be malicious. A simple attack program may be left running on an unattended machine in a public terminal room; it will look just like the usual logon screen, prompting for a user name and password. When an unsuspecting user does this, it will save the password somewhere in the system, reply "sorry, wrong password" and then vanish, invoking the genuine password program. The user will assume that he made a typing error the first time and think no more of it. This is why Windows NT has a "secure attention sequence," namely `ctrl-alt-del`, which is guaranteed to take you to a genuine password prompt. A facility that assures the user she's talking to a genuine system is called a *trusted path*.

If the whole terminal equipment is bogus, then of course all bets are off. We once caught a student installing modified keyboards in our public terminal room to capture passwords. When the attacker is prepared to take this much trouble, then all the `ctrl-alt-del` sequence achieves is to make his software design task simpler.

There have also been a few cases of criminals setting up false cash machines. In one famous case in Connecticut in 1993, the bad guys even bought genuine cash machines (on credit), installed them in a shopping mall, and proceeded to collect PINs and card details from unsuspecting bank customers who tried to use them [19]. Within a year, crooks in London had copied the idea, then enlarged on it by setting up a whole bogus bank branch [405]. Other cases have involved home-built cash machines, fitting false fronts over the front of genuine cash machines, or even replacing the card-operated door locks at the entrance to ATM facilities. Such attacks are even easier in countries where cards are used with PINs at the point of sale.

3.4.1.4 Technical Defeats of Password Retry Counters

Many kids find out that a bicycle combination lock can usually be broken in a few minutes by solving each ring in order of looseness. The same idea works against a number of computer systems. The PDP-10 TENEX operating system checked passwords one character at a time, and stopped as soon as one of them was wrong. This opened up a *timing attack*, whereby the attacker would repeatedly place a guessed password in memory at a suitable location, have it verified as part of a file access request, and wait to see how long it took to be rejected [493]. An error in the first character would be reported almost at once, an error in the second character would take a little longer to report, and in the third character a little longer still, and so on. So it was possible to guess the characters one after another, and instead of a password of N characters drawn from an alphabet of A characters taking $A^N/2$ guesses on average, it took $AN/2$. (Bear in mind that, in 30 years' time, all that might remain of the system you're building today is the memory of its more newsworthy security failures.)

A similar attack worked on one remote car-locking device: as soon as a wrong byte was transmitted from the key fob, the red telltale light on the receiver came on.

Password retry limits fail in other ways, too. With some smartcards, it has been possible to determine the customer PIN by trying each possible input value and looking at the card's power consumption, then issuing a reset if the input was wrong. The reason was that a wrong PIN caused a PIN retry counter to be decremented, and writing to the EEPROM memory that held this counter caused a current surge of several milliamps, which could be detected in time to reset the card before the write was complete [478].

3.4.2 Attacks on Password Storage

Passwords have often been vulnerable where they are stored. There was a horrendous bug in one operating system update in the 1980s: a user who entered a wrong password, and was told "sorry, wrong password" merely had to hit carriage return to get into the system anyway. This was spotted quickly, and a patch was shipped, but almost a hundred U.S. government systems in Germany were using unlicensed copies of the software and didn't get the patch, with the result that hackers were able to get in and steal information, which they are rumored to have sold to the KGB.

Another horrible programming error struck a U.K. bank, which issued all its customers with the same PIN by mistake. As the procedures for handling PINs were carefully controlled, no one in the bank got access to anyone's PIN other than his or her own, so the mistake wasn't spotted until after thousands of customer cards had been shipped.

3.4.2.1 Attacks via the Audit Trail

In systems that log failed password attempts, the log usually contains a large number of passwords, as users get the "username, password" sequence out of phase. If the logs are not well protected, then attacks become easy. Someone who sees an audit record of a failed login with a nonexistent user name of `e5gv, 8yp` can be 99 percent sure that this string is a password for one of the valid user names on the system.

3.4.2.2 One-Way Encryption

Password storage has also been a problem for some systems. Keeping a plaintext file of passwords can be dangerous. In MIT's Compatible Time Sharing System, `ctss` (a predecessor of Multics), it once happened that one person was editing the message of the day, while another was editing the password file. Because of a software bug, the two editor temporary files got swapped, with the result that everyone who logged on was greeted with a copy of the password file!

As a result of such incidents, passwords are often protected by encrypting them using a one-way algorithm, an innovation due to Roger Needham and Mike Guy. The password, when entered, is passed through a one-way function, and the user is logged on only if it matches a previously stored value.

Sometimes it isn't possible to protect a file of security information by one-way encryption, however, such as when this information must be processed in some way. The classic example is in GSM mobile phones, where each user has a cryptographic key on the home location register database. As this key is used to compute challenge-response pairs for authenticating users over the air, it is kept in the clear. (We will discuss the reasons for this design decision, and the possible alternatives, in Chapter 17).

3.4.2.3 Password Cracking

However, some systems that do use an encrypted password file make it *world readable* (Unix is the prime example—a design error now too well entrenched to change easily). This means that an opponent who can fetch this file can then try to break passwords offline using a dictionary; he encrypts the values in his dictionary and compares them with those in the file (an activity called a *dictionary attack*, or more colloquially, *password cracking*). NT is slightly better, but the password file can still be accessed by users who know what they're doing, and passwords may be passed to other systems (such as Netware, or earlier versions of NT) in old formats that use old, weak protection mechanisms for compatibility reasons.

Left to their own devices, people will use spouses' names, single letters, or even just hit Enter which gives an empty string as their password. So some systems require minimum password lengths, or even check user-entered passwords against a dictionary of bad choices. Still, designing a password quality enforcement mechanism is harder than one might think. Grampp and Morris's classic paper on Unix security [350] reports that after software became available that forced passwords to be at least six characters long and have at least one nonletter, they made a file of the 20 most common female names, each followed by a single digit. Of these 200 passwords, at least one was in use on each of several dozen machines they examined.

According to one report, when users were compelled to change their passwords, and prevented from using the previous few choices, they changed passwords rapidly to ex-

haust the history list and get back to their favorite password. A response, of forbidding password changes until after 15 days, meant that users couldn't change compromised passwords without help from the system administrator [603]. In my own experience, insisting on alphanumeric passwords and forcing a password change once a month led people to choose passwords such as julia03 for March, julia04 for April, and so on. So I am not at all convinced that demanding frequent password changes is a good idea.

A well-known study was conducted by Klein who gathered 25,000 Unix passwords in the form of encrypted password files and ran cracking software to guess them [460]. He found that 21 to 25 percent of passwords could be guessed, depending on the amount of effort put in. Dictionary words accounted for 7.4 percent, common names for 4 percent, combinations of user and account name 2.7 percent, and so on down a list of less-probable choices such as words from science fiction (0.4 percent) and sports terms (0.2 percent). Some of these were straightforward dictionary searches; others used patterns. For example, the algorithm for constructing combinations of user and account names would take an account `klone` belonging to the user Daniel V. Klein and try passwords such as `klone, klone, klone123, dvk, dvkdvk, leinad, neilk, DvkkvD`, and so on.

There are publicly available programs (`crack` for Unix and `L0phtcrack` for Windows [481]) that implement this kind of search. They can be used by system administrators to find bad passwords on their systems. They can just as easily be used by a bad guy who has got a copy of your password file. So password cracking is something to which you have to pay attention, especially if your system contains any machines running Unix or Linux. One way to use a program like `crack` is to filter user password choices; another is to use a custom program that understands language statistics and rejects passwords that are too likely to be chosen by others at random [98, 220]; another is to mix the two ideas using a suitable coding scheme [725].

3.4.3 Absolute Limits

Regardless of how well passwords are managed, there are often absolute limits imposed by the design of the operating system or other platform on which the system is built. For example, Unix systems limit the length of the password to eight characters (you can often enter more than this, but the ninth and subsequent characters are ignored). The effort required to try all possible passwords—the *total exhaust time*, in cryptanalytic jargon—is 96^8 or about 2^{52}; the average effort for a search is half of this. A well-financed government agency (or a well-organized hacker group, using PCs distributed across the Internet) could now break any encrypted password in a standard Unix password file.

This motivates more technical defenses against password cracking, including shadow passwords, that is, encrypted passwords hidden in a private file (most modern Unices), using an obscure mechanism to do the encryption (Novell), or using a secret key with the encryption (MVS). The strength of these mechanisms may vary.

For the above reasons, military system administrators often prefer to issue random passwords. This also lets the probability of password-guessing attacks be estimated and managed. For example, if L is the maximum password lifetime, R is login attempt rate, S is the size of the password space, then the probability that a password can be guessed in its lifetime is:

$$P = LR/S$$

This equation is taken from the U.S. Department of Defense password management guideline [242]. There are a couple of problems with this doctrine. First (the niggle) the password space can be completely exhausted, in which case $LR/S > 1$, and P isn't defined as a probability. Second (more serious) is that the attacker is not interested in guessing a password as much as getting acess to an account. So one has to take account of the number of users. If a large defense network has a million possible passwords and a million users, and the alarm goes off after three bad password attempts on any account, then the attack is to try one password for every single account. Thus, the quantity of real interest is the probability that the password space can be exhausted in the lifetime of the system at the maximum feasible password guess rate.

To take a concrete example, U.K. government systems tend to issue passwords that have been randomly selected with a fixed template of consonants, vowels, and numbers designed to make them easier to remember, such as CVCNCVCN (e.g., `fuR5_Eb8`). If passwords are not case-sensitive, the guess probability is only $21^4.5^2.10^2$, or about 2^{29}. So if an attacker could guess 100 passwords a second—perhaps distributed across 10,000 accounts on hundreds of machines on a network, so as not to raise the alarm—then he'd need about 5 million seconds, or two months, to get in.

In commercial systems, you can have a policy of simply blocking accounts after a number of false password attempts. If the threshold were three bad guesses in any one month, then with 10,000 accounts the maximum guess rate would be 30,000 passwords per month, and the rate at which guessing could be carried out undetectably would be much lower. But military system designers are reluctant to introduce account blocking, as it leaves them open to service denial attacks. As I mentioned in Section 3.3.2, an enemy who gets access to the network and enters enough wrong password guesses could freeze every account on the whole system.

3.5 Summary

Password management is one of the most important and yet most difficult design problems in many secure systems. As people get accounts on more and more systems, they reuse passwords in ways that expose serious vulnerabilities. But even where users operate in a controlled environment, things are by no means straightforward.

The ability to do offline password guessing more or less guarantees that an attacker will be able to compromise at least some accounts on any system, unless passwords are centrally assigned or filtered when users choose them. Where possible, one should stop offline guessing, for example, by keeping the password file secret. But systems such as Unix are often bought because of their large software base, and the software your customer wants to use may make changing the password mechanism difficult.

Critical questions to ask when designing a password system include not just whether people might reuse passwords, but also whether they need to be protected from each other, whether they can be trained and disciplined, and whether accounts can be frozen after a fixed number of bad guesses. You also have to consider whether attackers will target a particular account, or be happy with breaking any account on a machine or a network; and technical protection issues such as whether passwords can be snooped by malicious software, false terminals, or network eavesdropping.

Research Problems

I mentioned the lack of published empirical research. Although a little has been done, there's a lot more to do. For example, what are the best ways of enforcing user compliance with a password policy? There are some extreme solutions—such as issuing each user with a printed list of random passwords, each of which can be used once only—and these certainly work. But what can be done in applications where such drastic measures aren't justified?

Another problem, which straddles the borderline with security protocol design, is whether we can design interactive password systems that are better. There are various visual schemes and memorization schemes in the literature, and some early products: one system presents users with nine faces, only one of which is of a person they know; they have to pick the right face several times in a row to log on [223]. Other schemes present a table of numbers, and let the user do a secret computation using mental arithmetic. Designing such schemes is fairly easy; evaluating them is harder, as it involves elements of cryptology, psychology, and systems engineering.

An increasingly common mechanism is to ask for several pieces of security information rather than one. A call center might ask not just for your mother's maiden name, a password, and the amount of your last purchase, but also your dog's nickname and your favorite color. The underlying idea is that although an attacker might find out anything you know, it's much harder for him to find out everything you know. Again, such schemes need careful evaluation of their usability and effectiveness using the tools of applied psychology.

Further Reading

There isn't as much literature on passwords as one would like, despite the subject's importance. The papers by Bob Morris and Ken Thompson [561], Fred Grampp and Bob Morris [350], and Dan Klein [460], are the classics. The DoD guidelines are very influential [242].