*Information Security: Principles and Practice, Second Edition*
by Mark Stamp


# Part II

# Access Control

*Information Security: Principles and Practice, Second Edition*
by Mark Stamp


# Chapter 7

# Authentication

Guard: *Halt! Who goes there?*
Arthur: *It is I, Arthur, son of Uther Pendragon, from the castle of Camelot. King of the Britons, defeater of the Saxons, sovereign of all England!*
— *Monty Python and the Holy Grail*

*Then said they unto him, Say now Shibboleth: and he said Sibboleth: for he could not frame to pronounce it right. Then they took him, and slew him at the passages of Jordan: and there fell at that time of the Ephraimites forty and two thousand.*
— *Judges 12:6*

## 7.1 Introduction

We'll use the term *access control* as an umbrella for any security issues related to access of system resources. Within this broad definition, there are two areas of primary interest, namely, authentication and authorization.

Authentication is the process of determining whether a user (or other entity) should be allowed access to a system. In this chapter, our focus is on the methods used by humans to authenticate to local machines. Another type of authentication problem arises when the authentication information must pass through a network. While it might seem that these two authentication problems are closely related, in fact, they are almost completely different. When networks are involved, authentication is almost entirely an issue of security protocols. We'll defer our discussion of protocols to Chapters 9 and 10.

By definition, authenticated users are allowed access to system resources. However, an authenticated user is generally not given carte blanche access to all system resources. For example, we might only allow a privileged user—

such as an administrator—to install software on a system. How do we restrict the actions of authenticated users? This is the field of authorization, which is covered in the next chapter. Note that authentication is a binary decision—access is granted or it is not—while authorization is all about a more fine-grained set of restrictions on access to various system resources.

In security, terminology is far from standardized. In particular, the term access control is often used as a synonym for authorization. However, in our usage, access control is more broadly defined, with both authentication and authorization falling under the heading of access control. These two parts of access control can be summarized as follows.

- Authentication: Are you who you say you are?[1]
- Authorization: Are you allowed to do that?

## 7.2 Authentication Methods

In this chapter we address various methods that are commonly used to authenticate a human to a machine. That is, we want to convince a dumb machine that someone or something claiming to be Alice is indeed Alice and not, say, Trudy. That is, we want to answer the question, "Are you who you say you are?" Of course, we'd like to do this in as secure manner as possible.

A human can be authenticated to a machine based on any of the following[2] "somethings" [14].

- Something you know
- Something you have
- Something you are

A password is an example of "something you know." We'll spend some time discussing passwords, and in the process show that passwords represent a weak link in many modern information security systems.

An example of "something you have" is an ATM card or a smartcard. The "something you are" category is synonymous with the rapidly expanding field of biometrics. For example, today you can purchase a laptop that scans your thumbprint and uses the result for authentication. We'll discuss a few biometric methods later in this chapter. But first up are passwords.

[1] Try saying that three times, fast.

[2] Additional "somethings" are sometimes proposed. For example, one wireless access point authenticates a user by the fact that the user pushes a button on the device. This shows that the user has physical access to the device, and could be viewed as authentication by "something you do."

## 7.3 Passwords

*Your password must be at least 18770 characters*
*and cannot repeat any of your previous 30689 passwords.*
— Microsoft Knowledge Base Article 276304

An ideal password is something that you know, something that a computer can verify that you know, and something nobody else can guess—even with access to unlimited computing resources. We'll see that in practice it's difficult to even come close to this ideal.

Undoubtedly you are familiar with passwords. It's virtually impossible to use a computer today without accumulating a significant number of passwords. You probably log into your computer by entering a username and password, in which case you have obviously used a password. In addition, many other things that we don't call "password" act as passwords. For example, the PIN number used with an ATM card is, in effect, a password. And if you forget your password, a user-friendly website might authenticate you based on your social security number, your mother's maiden name, or your date of birth, in which case, these things are acting as passwords. A problem with such passwords is that they are often not secret.

If left to their own devices, users tend to select bad passwords, which makes password cracking surprisingly easy. In fact, we'll provide some basic mathematical arguments to show that it's inherently difficult to achieve security via passwords.

From a security perspective, a solution to the password problem would be to instead use randomly generated cryptographic keys. The work of cracking such a "password" would be equivalent to an exhaustive key search, in which case our passwords could be made at least as strong as our cryptography. The problem with such an approach is that humans must remember their passwords and we're not good at remembering randomly selected bits.

We're getting ahead of ourselves. Before discussing the numerous problems with passwords, we consider why passwords are so popular. Why is authentication based on "something you know" so much more popular than the more secure "somethings" (i.e., "something you have" and "something you are")? The answer, as always, is cost[3] and, secondarily, convenience. Passwords are free, while smartcards and biometric devices cost money. Also, it's more convenient for an overworked system administrator to reset a password than to provide a new smartcard or issue a user a new thumb.

[3]Students claim that when your Socratic author asks a question in his security class, the correct answer is invariably either "money" or "it depends."

### 7.3.1 Keys Versus Passwords

We've already claimed that cryptographic keys would solve the password problem. To see why this is so, let's compare keys to passwords. On the one hand, suppose our generic attacker, Trudy, is confronted with a 64-bit cryptographic key. Then there are $2^{64}$ possible keys, and, if the key was chosen at random (and assuming there is no shortcut attack), Trudy must on average try $2^{63}$ keys before she expects to find the correct one.

On the other hand, suppose Trudy is confronted with a password that is known to be eight characters long, with 256 possible choices for each character. Then there are $256^8 = 2^{64}$ possible passwords. At first glance, cracking such passwords might appear to be equivalent to the key search problem. Unfortunately (or, from Trudy's perspective, fortunately) users don't select passwords at random, because users must remember their passwords. As a result, a user is far more likely to choose an 8-character dictionary word such as

```
password
```

than, say,

```
kf&Yw!a[
```

So, in this case Trudy can make far fewer than $2^{63}$ guesses and have a high probability of successfully cracking a password. For example, a carefully selected dictionary of $2^{20} \approx 1,000,000$ passwords would likely give Trudy a reasonable probability of cracking a given password. On the other hand, if Trudy attempted to find a randomly generated 64-bit key by trying only $2^{20}$ possible keys, her chance of success would be a mere $2^{20}/2^{64} = 1/2^{44}$, or less than 1 in 17 trillion. The bottom line is that the non-randomness of password selection is at the root of the problems with passwords.

### 7.3.2 Choosing Passwords

Not all passwords are created equal. For example, everyone would probably agree that the following passwords are weak:

- `Frank`
- `Pikachu`
- `10251960`
- `AustinStamp`

especially if your name happens to be Frank, or Austin Stamp, or your birthday is on 10/25/1960.

Security often rests on passwords and, consequently, users should have passwords that are difficult to guess. However, users must be able to remember their passwords. With that in mind, are the following passwords better than the weak passwords above?

- `jfIej(43j-EmmL+y`
- `09864376537263`
- `POkemON`
- `FSa7Yago`

The first password, `jfIej(43j-EmmL+y`, would certainly be difficult for Trudy to guess, but it would also be difficult for Alice to remember. Such a password is likely to end up on the proverbial post-it note stuck to the front of Alice's computer. This could make Trudy's job much easier than if Alice had selected a "less secure" password.

The second password on the list above is also probably too much for most users to remember. Even the highly trained U.S. military personal responsible for launching nuclear missiles are only required to remember 12-digit firing codes [14].

The password `POkemON` might be difficult to guess, since it's not a standard dictionary word due to the digits and the upper case letters. However, if the user were known to be a fan of Pokémon, this password might be relatively easy prey.

The final password, `FSa7Yago`, might appear to reside in the difficult to guess, but too difficult to remember category. However, there is a trick to help the user remember it—it's based on a *passphrase*. That is, `FSa7Yago` is derived from the phrase "four score and seven years ago." Consequently, this password should be relatively easy for Alice to remember, and yet relatively difficult for Trudy to guess.

An interesting password experiment is described in [14]. Users were divided into three groups, and given the following advice regarding password selection:

- Group A — Select passwords consisting of at least six characters, with at least one non-letter. This is fairly typical password selection advice.
- Group B — Select passwords based on passphrases.
- Group C — Select passwords consisting of eight randomly selected characters.

The experimenters tried to crack the resulting passwords for each of the three groups. The results were as follows:

- Group A — About 30% of passwords were easy to crack. Users in this group found their passwords easy to remember.
- Group B — About 10% of the passwords were cracked, and, as with users in Group A, users in this group found their passwords easy to remember.
- Group C — About 10% of the passwords were cracked. Not surprisingly, the users in this group found their passwords difficult to remember.

These results clearly indicate that passphrases provide the best option for password selection, since the resulting passwords are relatively difficult to crack yet easy to remember.

This password experiment also demonstrated that user compliance is hard to achieve. In each of groups A, B, and C, about one-third of the users did not comply with the instructions. Assuming that non-compliant users tend to select passwords similar to Group A, about one-third of these passwords would be easy to crack. The bottom line is that nearly 10% of passwords are likely to be easy to crack, regardless of the advice given.

In some situations, it makes sense to assign passwords, and if this is the case, noncompliance with the password policy is a non-issue. The trade-off here is that users are likely to have a harder time remembering assigned passwords as compared to passwords they select themselves.

Again, if users are allowed to choose passwords, then the best advice is to choose passwords based on passphrases. In addition, system administrators should use a password-cracking tool to test for weak passwords, since attackers certainly will.

It is also sometimes suggested that periodic password changes should be required. However, users can be very clever at avoiding such requirements, invariably to the detriment of security. For example, Alice might simply "change" her password without changing it. In response to such users, the system could remember, say, five previous passwords. But a clever user like Alice will soon learn that she can cycle through five password changes and then reset her password to its original value. Or, if Alice is required to choose a new password each month she might select, say, `frank01` in January, `frank02` in February, and so on. Forcing reluctant users to choose reasonably strong passwords is not as simple as it might seem.

### 7.3.3 Attacking Systems via Passwords

Suppose that Trudy is an outsider, that is, she has no access to a particular system. A common attack path for Trudy would be

$$\text{outsider} \longrightarrow \text{normal user} \longrightarrow \text{administrator}.$$

In other words, Trudy will initially seek access to any account on the system and then attempt to upgrade her level of privilege. In this scenario, one weak password on a system—or in the extreme, one weak password on an entire network—could be enough for the first stage of the attack to succeed. The bottom line is that one weak password may be one too many.

Another interesting issue concerns the proper response when attempted password cracking is detected. For example, systems often lock users out after three bad passwords attempts. If this is the case, how long should the system lock? Five seconds? Five minutes? Until the administrator manually resets the service? Five seconds might be insufficient to deter an automated attack. If it takes more than five seconds for Trudy to make three password guesses for every user on the system, then she could simply cycle through all accounts, making three guesses on each. By the time she returns to a particular user's account, more than five seconds will have elapsed and she will be able to make three more guesses without any delay. On the other hand, five minutes might open the door to a denial of service attack, where Trudy is able to lock accounts indefinitely by periodically making three password guesses on an account. The correct answer to this dilemma is not readily apparent.

### 7.3.4 Password Verification

Next, we consider the important issue of verifying that an entered password is correct. For a computer to determine the validity of a password, it must have something to compare against. That is, the computer must have access to the correct password in some form. But it's probably a bad idea to simply store the actual passwords in a file, since this would be a prime target for Trudy. Here, as in many other areas in information security, cryptography provides a sound solution.

It might be tempting to encrypt the password file with a symmetric key. However, to verify passwords, the file must be decrypted, so the decryption key must be as accessible as the file itself. Consequently, if Trudy can steal the password file, she can probably steal the key as well. Consequently, encryption is of little value here.

So, instead of storing raw passwords in a file or encrypting the password file, it's more secure to store hashed passwords. For example, if Alice's password is `FSa7Yago`, we could store

$$y = h(\texttt{FSa7Yago})$$

in a file, where $h$ is a secure cryptographic hash function. Then when someone claiming to be Alice enters a password $x$, it is hashed and compared to $y$, and if $y = h(x)$ then the entered password is assumed to be correct and the user is authenticated.

The advantage of hashing passwords is that if Trudy obtains the password file, she does not obtain the actual passwords—instead she only has the hashed passwords. Note that we are relying on the one-way property of cryptographic hash functions to protect the passwords. Of course, if Trudy knows the hash value $y$, she can conduct a forward search attack by guessing likely passwords $x$ until she finds an $x$ for which $y = h(x)$, at which point she will have cracked the password. But at least Trudy has work to do after she has obtained the password file.

Suppose Trudy has a dictionary containing $N$ common passwords, say,

$$d_0, d_1, d_2, \ldots, d_{N-1}.$$

Then she could precompute the hash of each password in the dictionary,

$$y_0 = h(d_0), y_1 = h(d_1), \ldots, y_{N-1} = h(d_{N-1}).$$

Now if Trudy gets access to a password file containing hashed passwords, she only needs to compare the entries in the password file to the entries in her precomputed dictionary of hashes. Furthermore, the precomputed dictionary could be reused for each password file, thereby saving Trudy the work of recomputing the hashes. And if Trudy is feeling particularly generous, she could post her dictionary of common passwords and their corresponding hashes online, saving all other attackers the work of computing these hashes. From the good guy's point of view, this is a bad thing, since the work of computing the hashes has been largely negated. Can we prevent this attack, or at least make Trudy's job more difficult?

Recall that to prevent a forward search attack on public key encryption, we append random bits to the message before encrypting. We can accomplish a similar effect with passwords by appending a non-secret random value, known as a *salt*, to each password before hashing. A password salt is analogous to the initialization vector, or IV, in, say, cipher block chaining (CBC) mode encryption. Whereas an IV is a non-secret value that causes identical plaintext blocks to encrypt to different ciphertext values, a salt is a non-secret value that causes identical password to hash to different values.

Let $p$ be a newly entered password. We generate a random salt value $s$ and compute $y = h(p, s)$ and store the pair $(s, y)$ in the password file. Note that the salt $s$ is no more secret than the hash value. Now to verify an entered password $x$, we retrieve $(s, y)$ from the password file, compute $h(x, s)$, and compare this result with the stored value $y$. Note that salted password verification is just as easy as it was in the unsalted case. But Trudy's job has become much more difficult. Suppose Alice's password is hashed with salt value $s_a$ and Bob's password is hashed with salt value $s_b$. Then, to test Alice's password using her dictionary of common passwords, Trudy must compute the hash of each word in her dictionary with salt value $s_a$, but to attack

Bob's password, Trudy must recompute the hashes using salt value $s_b$. For a password file with $N$ users, Trudy's work has just increased by a factor of $N$. Consequently, a precomputed file of hashed passwords is no longer useful for Trudy. She can't be pleased with this turn of events.[4]

### 7.3.5 Math of Password Cracking

Now we'll take a look at the math behind password cracking. Throughout this section, we'll assume that all passwords are eight characters in length and that there are 128 choices for each character, which implies there are

$$128^8 = 2^{56}$$

possible passwords. We'll also assume that passwords are stored in a password file that contains $2^{10}$ hashed passwords, and that Trudy has a dictionary of $2^{20}$ common passwords. From experience, Trudy expects that any given password will appear in her dictionary with a probability of about $1/4$. Also, work is measured by the number of hashes computed. Note that comparisons are free—only hash calculations count as work.

Under these assumptions, we'll determine the probability of successfully cracking a password in each of the following four cases.

I. Trudy wants to determine Alice's password (perhaps Alice is the administrator). Trudy does not use her dictionary of likely passwords.

II. Trudy wants to determine Alice's password. Trudy does use her dictionary of common passwords.

III. Trudy will be satisfied to crack any password in the password file, without using her dictionary.

IV. Trudy wants to find any password in the hashed password file, using her dictionary.

In each case, we'll consider both salted and unsalted passwords.

**Case I**: Trudy has decided that she wants to crack Alice's password. Trudy, who is somewhat absent-minded, has forgotten that she has a password dictionary available. Without a dictionary of common passwords, Trudy has no choice other than a brute force approach. This is precisely equivalent to an exhaustive key search and hence the expected work is

$$2^{56}/2 = 2^{55}.$$

[4]Salting password hashes is as close to a free lunch as you'll come in information security. Maybe the connection with a free lunch is why it's called a salt?

The result here is the same whether the passwords are salted or not, unless someone has precomputed, sorted, and stored the hashes of all possible passwords. If the hashes of all passwords are already known, then in the unsalted case, there is no work at all—Trudy simply looks up the hash value and finds the corresponding password. But, if the passwords are salted, there is no benefit to having the password hashes. In any case, precomputing all possible password hashes is a great deal of work, so for the remainder of this discussion, we'll assume this is infeasible.

**Case II**: Trudy again wants to recover Alice's password, and she is going to use her dictionary of common passwords. With probability 1/4, Alice's password is in Trudy's dictionary. Suppose the passwords are salted. Furthermore, suppose Alice's password is in Trudy's dictionary. Then Trudy would expect to find Alice's password after hashing half of the words in the dictionary, that is, after $2^{19}$ tries. With probability 3/4 the password is not in the dictionary, in which case Trudy would expect to find it after about $2^{55}$ tries. Combining these cases gives Trudy an expected work of

$$\frac{1}{4}\left(2^{19}\right) + \frac{3}{4}\left(2^{55}\right) \approx 2^{54.6}.$$

Note that the expected work here is almost the same as in Case I, where Trudy did not use her dictionary. However, in practice, Trudy would simply try all the words in her dictionary and quit if she did not find Alice's password. Then the work would be at most $2^{20}$ and the probability of success would be 1/4.

If the passwords are unsalted, Trudy could precompute the hashes of all $2^{20}$ passwords in her dictionary. Then this small one-time work could be amortized over the number of times that Trudy uses this attack. That is, the larger the number of attacks, the smaller the average work per attack.

**Case III**: In this case, Trudy will be satisfied to determine any of the 1024 passwords in the hashed password file. Trudy has again forgotten about her password dictionary.

Let $y_0, y_1, \ldots, y_{1023}$ be the password hashes. We'll assume that all $2^{10}$ passwords in the file are distinct. Let $p_0, p_1, \ldots, p_{2^{56}-1}$ be a list of all $2^{56}$ possible passwords. As in the brute force case, Trudy needs to make $2^{55}$ distinct comparisons before she expects to find a match.

If the passwords are not salted, then Trudy can compute $h(p_0)$ and compare it with each $y_i$, for $i = 0, 1, 2, \ldots, 1023$. Next she computes $h(p_1)$ and compares it with all $y_i$ and so on. The point here is that each hash computation provides Trudy with $2^{10}$ comparisons. Since work is measured in terms of hashes, not comparisons, and $2^{55}$ comparisons are needed, the expected work is

$$2^{55}/2^{10} = 2^{45}.$$

Now suppose the passwords are salted. Let $s_i$ denote the salt value corresponding to hash password $y_i$. Then Trudy computes $h(p_0, s_0)$ and compares it with $y_0$. Next, she computes $h(p_0, s_1)$ and compares it with $y_1$, she computes $h(p_0, s_2)$ and compares it with $y_2$, and she continues in this manner up to $h(p_0, s_{1023})$. Then Trudy must repeat this entire process with password $p_1$ in place of $p_0$, and then with password $p_2$ and so on. The bottom line is that each hash computation only yields one comparison and consequently the expected work is $2^{55}$, which is the same as in Case I above.

This case illustrates the benefit of salting passwords. However, Trudy has not made use of her password dictionary, which is unrealistic.

**Case IV**: Finally, suppose that Trudy will be satisfied to recover any one of the 1024 passwords in the hashed password file, and she will make use of her password dictionary. First, note that the probability that at least one of the 1024 passwords in the file appears in Trudy's dictionary is

$$1 - \left(\frac{3}{4}\right)^{1024} \approx 1.$$

Therefore, we can safely ignore the case where no password from the file is in Trudy's dictionary.

If the passwords are not salted, then Trudy could simply hash all password in her dictionary and compare the results to all 1024 hashes in the password file. Since we are certain that at least one of these passwords is in the dictionary, Trudy's work is $2^{20}$ and she is assured of finding at least one password. However, if Trudy is a little more clever, she can greatly reduce this meager work factor. Again, we can safely assume that at least one of the passwords is in Trudy's dictionary. Consequently, Trudy only needs to make about $2^{19}$ comparisons—half the size of her dictionary—before she expects to find a password. As in Case III, each hash computation yields $2^{10}$ comparisons, so the expected work is only

$$2^{19}/2^{10} = 2^9.$$

Finally, note that in this unsalted case, if the hashes of the dictionary passwords have been precomputed, no additional work is required to recover one (or more) passwords. That is, Trudy simply compares the hashes in the file to the hashes of her dictionary passwords and, in the process, she recovers any passwords that appear in her dictionary.

Now we consider the most realistic case—Trudy has a dictionary of common passwords, she will be happy to recover any password from the password file, and the passwords in the file are salted. For this case, we let $y_0, y_1, \ldots, y_{1023}$ be the password hashes and $s_0, s_1, \ldots, s_{1023}$ be the corresponding salt values. Also, let $d_0, d_1, \ldots, d_{2^{20}-1}$ be the dictionary words. Suppose that Trudy first computes $h(d_0, s_0)$ and compares it to $y_0$, then she

compute $h(d_1, s_0)$ and compares it to $y_0$, then she compute $h(d_2, s_0)$ and compares it to $y_0$, and so on. That is, Trudy first compares $y_0$ to all of her (hashed) dictionary words. Of course, she must use salt $s_0$ for these hashes. If she does not recover the password corresponding to $y_0$, then she repeats the process using $y_1$ and $s_1$, and so on.

Note that if $y_0$ is in the dictionary (which has probability 1/4), Trudy expects to find it after about $2^{19}$ hashes, while if it is not in the dictionary (probability 3/4) Trudy will compute $2^{20}$ hashes. If Trudy finds $y_0$ in the dictionary, then she's done. If not, Trudy will have computed $2^{20}$ hashes before she moves on to consider $y_1$. Continuing in this manner, we find that the expected work is about

$$\frac{1}{4}\left(2^{19}\right) + \frac{3}{4}\cdot\frac{1}{4}\left(2^{20}+2^{19}\right) + \left(\frac{3}{4}\right)^2\frac{1}{4}\left(2\cdot 2^{20}+2^{19}\right) + \cdots + \left(\frac{3}{4}\right)^{1023}\frac{1}{4}\left(1023\cdot 2^{20}+2^{19}\right) < 2^{22}.$$

This is somewhat disappointing, since it shows that, for very little work, Trudy can expect to crack at least one password.

It can be shown (see Problems 24 and 25) that, under reasonable assumptions, the work needed to crack a (salted) password is approximately equal to size of the dictionary divided by the probability that a given password is in the dictionary. In our example here, the size of the dictionary is $2^{20}$ while the probability of finding a password is 1/4. So, the expected work should be about

$$\frac{2^{20}}{1/4} = 2^{22}$$

which is consistent with the calculation above. Note that this approximation implies that we can increase Trudy's work by forcing her to have a larger dictionary or by decreasing her probability of success (or both), which makes intuitive sense. Of course, the obvious way to accomplish this is to choose passwords that are harder to guess.

The inescapable conclusion is that password cracking is too easy, particularly in situations where one weak password is sufficient to break the security of an entire system. Unfortunately, when it comes to passwords, the numbers strongly favor the bad guys.

### 7.3.6 Other Password Issues

As bad as it is, password cracking is only the tip of the iceberg when it comes to problems with passwords. Today, most users need multiple passwords, but users can't (or won't) remember a large number of passwords. This results in a significant amount of password reuse, and any password is only as secure

as the least secure place it's used. If Trudy finds one of your passwords, she would be wise to try it (and slight variations of it) in other places where you use a password.

Social engineering is also a major concern with passwords.[5] For example, if someone calls you, claiming to be a system administrator who needs your password to correct a problem with your account, would you give away your password? According to a recent survey, 34% of users will give away their password if you ask for it, and the number increases to 70% if you offer a candy bar as incentive [232].

Keystroke logging software and similar spyware are also serious threats to password-based security [22]. The failure to change default passwords is a major source of attacks as well [339].

An interesting question is, who suffers from bad passwords? The answer is that it depends. If you choose your birthday for your ATM PIN number, only you stand to lose.[6] On the other hand, if you choose a weak password at work, the entire company stands to lose. This explains why banks usually let users choose any PIN number they desire for their ATM cards, but companies generally try to force users to select reasonably strong passwords.

There are many popular password cracking tools including L0phtCrack [2] (for Windows) and John the Ripper [157] (for Unix). These tools come with preconfigured dictionaries, and it is easy to produce customized dictionaries. These are good examples of the types of tools that are available to hackers.[7] Since virtually no skill is required to leverage these powerful tools, the door to password cracking is open to all, regardless of ability.

Passwords are one of the most severe real-world security problems today, and this is unlikely to change any time soon. The bad guys clearly have the advantages when it comes to passwords. In the next section, we'll look at biometrics, which—together with smartcards ad similar devices—are often touted as the best way to escape from the multitude of problems inherent with passwords.

[5]Actually, social engineering is a major concern in all aspects of information security where humans play a role. Your all-too-human author heard a talk about penetration testing, where the tester was paid to probe the security of a major corporation. The tester lied and forged a (non-digital) signature to obtain entry into corporate headquarters, where he posed as a system administrator trainee. Secretaries and other employees were more than happy to accept "help" from this fake SA trainee. As a result, the tester claimed to have obtained almost all of the company's intellectual property (including such sensitive information as the design of nuclear power plants) within two days. This attack consisted almost entirely of social engineering.

[6]Perhaps the bank will lose too, but only if you live in the United States and you have a very good lawyer.

[7]Of course, almost every hacker tool has legitimate uses. For example, password cracking tools are valuable for system administrators, since they can use these tools to test the strength of the passwords on their system.

## 7.4 Biometrics

*You have all the characteristics of a popular politician:*
*a horrible voice, bad breeding, and a vulgar manner.*
— Aristophanes

Biometrics represent the "something you are" method of authentication or, as Schneier so aptly puts it, "you are your key" [260]. There are many different types of biometrics, including such long-established methods as fingerprints. Recently, biometrics based on speech recognition, gait (walking) recognition, and even a digital doggie (odor recognition) have been developed. Biometrics are currently a very active topic for research [151, 176].

In the information security arena, biometrics are seen as a more secure alternative to passwords. For biometrics to be a practical replacement for passwords, cheap and reliable systems are needed. Today, usable biometric systems exist, including laptops using thumbprint authentication, palm print systems for secure entry into restricted facilities, the use of fingerprints to unlock car doors, and so on. But given the potential of biometrics—and the well-known weaknesses of password-based authentication—it's perhaps surprising that biometrics are not more widely used.

An ideal biometric would satisfy all of the following:

- Universal — A biometric should apply to virtually everyone. In reality, no biometric applies to everyone. For example, a small percentage of people do not have readable fingerprints.
- Distinguishing — A biometric should distinguish with virtual certainty. In reality, we can't hope for 100% certainty, although, in theory, some methods can distinguish with very low error rates.
- Permanent — Ideally, the physical characteristic being measured should never change. In practice, it's sufficient if the characteristic remains stable over a reasonably long period of time.
- Collectable — The physical characteristic should be easy to collect without any potential to cause harm to the subject. In practice, collectability often depends heavily on whether the subject is cooperative or not.
- Reliable, robust, and user-friendly — These are just some of the additional real-world considerations for a practical biometric system. Some biometrics that have shown promise in laboratory conditions have subsequently failed to deliver similar performance in practice.

Biometrics are also applied in various *identification* problems. In the identification problem we are trying to answer the question "Who are you?," while

for the authorization problem, we want to answer the question, "Are you who you say you are?" That is, in identification, the goal is to identify the subject from a list of many possible subjects. This occurs, for example, when a suspicious fingerprint from a crime scene is sent to the FBI fingerprint database for comparison with all of the millions of fingerprint records currently on file.

In the identification problem, the comparison is one-to-many whereas for authentication, the comparison is one-to-one. For example, if someone claiming to be Alice uses a thumbprint mouse biometric, the captured thumbprint image is only compared with the stored thumbprint of Alice. The identification problem is inherently more difficult and subject to a much higher error rate due to the larger number of comparisons that must be made. That is, each comparison carries with it a probability of an error, so the more comparisons required, the higher the error rate.

There are two phases to a biometric system. First, there is an *enrollment phase*, where subjects have their biometric information gathered and entered into a database. Typically, during this phase very careful measurement of the pertinent physical information is required. Since this is one-time work (per subject), it's acceptable if the process is slow and multiple measurements are required. In some fielded systems, enrollment has proven to be a weak point since it may be difficult to obtain results that are comparable to those obtained under laboratory conditions.

The second phase in a biometric system is the *recognition phase*. This occurs when the biometric detection system is used in practice to determine whether (for the authentication problem) to authenticate the user or not. This phase must be quick, simple, and accurate.

We'll assume that subjects are cooperative, that is, they're willing to have the appropriate physical characteristic measured. This is a reasonable assumption in the authentication case, since authentication is generally required for access to certain information resources or for entry into an otherwise restricted area.

For the identification problem, it is often the case that subjects are uncooperative. For example, consider a facial recognition system used for identification. Las Vegas casinos use such systems to detect known cheaters as they attempt to enter a casino [300]. Another fanciful proposed use of facial recognition is to spot terrorists in airports.[8] In such cases, the enrollment conditions may be far from ideal, and in the recognition phase, the subjects are certainly uncooperative as they likely do everything possible to avoid detection. Of course, uncooperative subjects can only serve to make the underlying biometric problem more difficult. For the remainder of this discussion we'll focus on the authentication problem and we'll assume that the subjects are cooperative.

[8]Apparently, terrorists are welcome in casinos, as long as they don't cheat.

### 7.4.1 Types of Errors

There are two types of errors that can occur in biometric recognition. Suppose Bob poses as Alice and the system mistakenly authenticates Bob as Alice. The rate at which such misauthentication occurs is the *fraud rate*. Now suppose that Alice tries to authenticate as herself, but the system fails to authenticate her. The rate at which this type of error occurs is the *insult rate* [14].

For any biometric, we can decrease the fraud or insult rate at the expense of the other. For example, if we require a 99% voiceprint match, then we can obtain a low fraud rate, but the insult rate will be high, since a speaker's voice will naturally change slightly from time to time. On the other hand, if we set the threshold at a 30% voiceprint match, the the fraud rate will likely be high, but the system will have a low insult rate.

The *equal error rate* is the rate for which the fraud and insult rates are the same. That is, the parameters of the system are adjusted until the fraud rate and insult rate are precisely in balance. This is a useful measure for comparing different biometric systems.

### 7.4.2 Biometric Examples

In this section, we'll briefly discuss three common biometrics. First, we'll consider fingerprints, which, in spite of their long history, are relative newcomers in computing applications. Then we'll discuss palm prints and iris scans.

#### 7.4.2.1 Fingerprints

Fingerprints were used in ancient China as a form of signature, and they have served a similar purpose at other times in history. But the use of fingerprints as a scientific form of identification is a much more recent phenomenon.

A significant analysis of fingerprints occurred in 1798 when J. C. Mayer suggested that fingerprints might be unique. In 1823, Johannes Evangelist Purkinje discussed nine fingerprint patterns, but this work was a biological treatise and did not suggest using fingerprints as a form of identification. The first modern use of fingerprints for identification occurred in 1858 in India, when Sir William Hershel used palm prints and fingerprints as forms of signatures on contracts.

In 1880, Dr. Henry Faulds published an article in *Nature* that discussed the use of fingerprints for identification purposes. In Mark Twain's *Life on the Mississippi*, which was published in 1883, a murderer is identified by a fingerprint. However, the widespread use of fingerprinting only became possible in 1892 when Sir Francis Galton developed a classification system based on "minutia" that enabled efficient searching, and he verified that fingerprints do not change over time [188].

Examples of the different types of minutia in Galton's classification system appear in Figure 7.1. Galton's system allowed for an efficient solution to the identification problem in the pre-computer era.[9]


Figure 7.1: Examples of Galton's Minutia

Today, fingerprints are routinely used for identification, particularly in criminal cases. It is interesting to note that the standard for determining a match varies widely. For example, in Britain fingerprints must match in 16 points, whereas in the United States, no fixed number of points are required to match.[10]

A fingerprint biometric works by first capturing an image of the fingerprint. The image is then enhanced using various image-processing techniques, and various points are identified and extracted from the enhanced image. This process is illustrated in Figure 7.2.

Figure 7.2: Automatic Extraction of Minutia

The points extracted by the biometric system are compared in a manner that is somewhat analogous to the manual analysis of fingerprints. For authentication, the extracted points are compared with the claimed user's stored

[9]Fingerprints were classified into one of 1024 "bins." Then, given a fingerprint from an unknown subject, a binary search based on the minutia quickly focused the effort of matching the print on one of these bins. Consequently, only a very small subset of recorded fingerprints needed to be carefully compared to the unknown fingerprint.

[10]This is a fine example of the way that the U.S. generously ensures full employment for lawyers—they can always argue about whether fingerprint evidence is admissible or not.

information, which was previously captured during the enrollment phase. The system then determines whether a statistical match occurs, with some predetermined level of confidence. This fingerprint comparison process is illustrated in Figure 7.3.

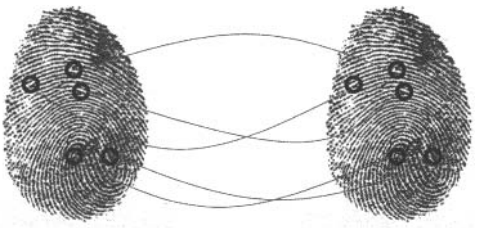

Figure 7.3: Minutia Comparison

#### 7.4.2.2 Hand Geometry

Another popular biometric is hand geometry, which is particularly popular for entry into secure facilities [138, 256]. In this system, the shape of the hand is carefully measured, including the width and length of the hand and fingers.[11] The paper [152] describes 16 such measurements, of which 14 are illustrated in Figure 7.4 (the other two measure the thickness of the hand). Human hands are not nearly as unique as fingerprints, but hand geometry is easy and quick to measure, while being sufficiently robust for many authentication uses. However, hand geometry would probably not be suitable for identification, since the number of false matches would be high.

One advantage of hand geometry systems is that they are fast, taking less than one minute in the enrollment phase and less than five seconds in the recognition phase. Another advantage is that human hands are symmetric, so if the enrolled hand is, say, in a cast, the other hand can be used by placing it palm side up. Some disadvantages of hand geometry include that it cannot be used on the young or the very old, and, as we'll discuss in a moment, the system has a relatively high equal error rate.

#### 7.4.2.3 Iris Scan

A biometric that is, in theory, one of the best for authentication is the iris scan. The development of the iris (the colored part of the eye) is chaotic, which implies that minor variations lead to large differences. There is little or no genetic influence on the iris pattern, so that the measured pattern

[11] Note that palm print systems do not read your palm. For that, you'll have to see your local chiromancer.

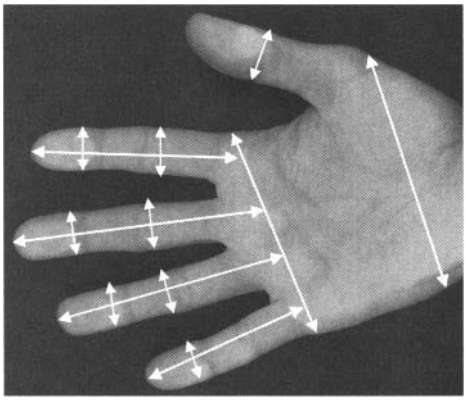

Figure 7.4: Hand Geometry Measurements

is uncorrelated for identical twins and even for the two eyes of one individual. Another desirable property is that the pattern is stable throughout a lifetime [149].

The development of iris scan technology is relatively new. In 1936, the idea of using the human iris for identification was suggested by Frank Burch. In the 1980s, the idea resurfaced in James Bond films, but it was not until 1986 that the first patents appeared—a sure sign that people foresaw money to be made on the technology. In 1994, John Daugman, a researcher at Cambridge University, patented what is generally accepted as the best approach currently available [76].

Iris scan systems require sophisticated equipment and software. First, an automated iris scanner locates the iris. Then a black and white photo of the eye is taken. The resulting image is processed using a two-dimensional wavelet transform, the result of which is a 256-byte (that is, 2048-bit) iris code.

Two iris codes are compared based on the Hamming distance between the codes. Suppose that Alice is trying to authenticate using an iris scan. Let $x$ be the iris code computed from Alice's iris in the recognition phase, while $y$ is Alice's iris code stored in the scanner's database, which was gathered during the enrollment phase. Then $x$ and $y$ are compared by computing the distance $d(x, y)$ defined by

$$d(x, y) = \frac{\text{number of non-match bits}}{\text{number of bits compared}}. \tag{7.1}$$

For example, $d(0010, 0101) = 3/4$ and $d(101111, 101001) = 1/3$.

For an iris scan, $d(x, y)$ is computed on the 2048-bit iris code. A perfect match yields $d(x, y) = 0$, but we can't expect perfection in practice. Under

laboratory conditions, for the same iris the expected distance is 0.08, and for different irises the expect distance is 0.50. The usual thresholding scheme is to accept the comparison as a match if the distance is less than 0.32 and otherwise consider it a non-match [76]. An image of an iris appears in Figure 7.5.

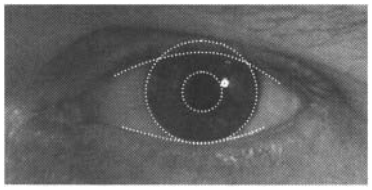

Figure 7.5: An Iris Scan

Define the *match* cases to be those where, for example, Alice's data from the enrollment phase is compared to her scan data from the recognition phase. Define the *no-match* cases to be when, for example, Alice's enrollment data is compared to Bob's recognition phase data (or vice versa). Then the left histogram in Figure 7.6 represents match data, while the right histogram represents no-match data. Note that the match data provides information relevant to the insult rate, whereas the no-match data provides information relevant to the fraud rate.

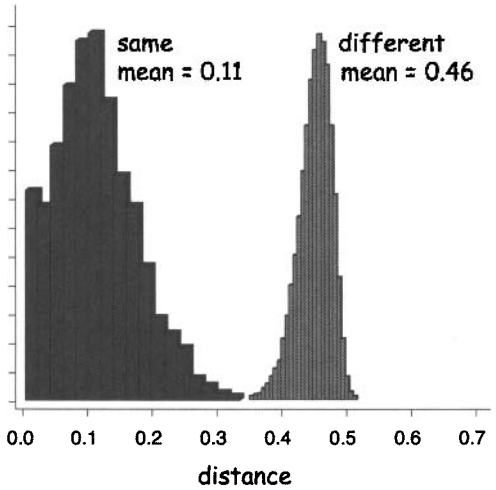


Figure 7.6: Histogram of Iris Scan Results [149]

The iris scan is often cited as the ultimate biometric for authentication. The histogram in Figure 7.6, which is based on 2.3 million comparisons, tends

to support this view, since the overlapping region between the "same" (match) and "different" (no-match) cases appears to be virtually nonexistent. Note that the overlap represents the region where an error can occur. In reality, there is some overlap between the histograms in in Figure 7.6, but the overlap is extremely small.

The iris scan distances for the match data in Table 7.1 provide a more detailed view of the same histogram marked as "same" in Figure 7.6. From Figure 7.6, we see that the equal error rate (which corresponds to the crossover point between the two graphs) occurs somewhere near distance 0.34. From Table 7.1, this implies an equal error rate of about $10^{-5}$. For this biometric, we would certainly be willing to tolerate a slightly higher insult rate since that would further reduce the fraud rate. Hence, the typical threshold used is 0.32, as mentioned above.

Table 7.1: Iris Scan Match Scores and Error Rates [149]

| Score | Probability |
|---|---|
| 0.29 | 1 in $1.3 \times 10^{10}$ |
| 0.30 | 1 in $1.5 \times 10^{9}$ |
| 0.31 | 1 in $1.8 \times 10^{8}$ |
| 0.32 | 1 in $2.6 \times 10^{7}$ |
| 0.33 | 1 in $4.0 \times 10^{6}$ |
| 0.34 | 1 in $6.9 \times 10^{5}$ |
| 0.35 | 1 in $1.3 \times 10^{5}$ |

Is it possible to attack an iris-scanning system? Suppose Bob has a good photo of Alice's eye. Then he can claim to be Alice and try to use the photo to trick the system into authenticating him as Alice. This attack is not at all far fetched. In fact, an Afghan woman whose photo appeared on a famous *National Geographic* magazine cover in 1984 was positively identified 17 years later by comparing her then-current iris scan with an iris scan taken from the 1984 photo. The woman had never seen the magazine, but she did recall being photographed. The magazine cover with the woman's photo and the fascinating story of finding this person after years of war and chaos in Afghanistan can be found at [28].

To prevent attacks based on a photo, an iris-scanning system could first shine a light on the "eye" and verify that the pupil contracts before proceeding with the iris scan. While this eliminates an attack that relies on a static photo, it also might significantly increase the cost of the system. Given that biometrics are in competition with passwords, and passwords are free, cost is always an issue.

### 7.4.3 Biometric Error Rates

Recall that the equal error rate—the point at which the fraud rate equals the insult rate—is generally considered the best measure for comparing different biometric systems. The equal error rates for several popular biometrics are given in Table 7.2.

Table 7.2: Biometric Equal Error Rates [32]

| Biometric | Equal error rate |
|---|---|
| fingerprint | $2.0 \times 10^{-3}$ |
| hand geometry | $2.0 \times 10^{-3}$ |
| voice recognition | $2.0 \times 10^{-2}$ |
| iris scan | $7.6 \times 10^{-6}$ |
| retina scan | $1.0 \times 10^{-7}$ |
| signature recognition | $2.0 \times 10^{-2}$ |

For fingerprint biometric systems, the equal error rate may seem high. However, most fingerprint biometrics are relatively cheap devices that do not achieve anything near the theoretical potential for fingerprint matching. On the other hand, hand geometry systems are relatively expensive and sophisticated devices, so they probably do achieve something close to the theoretical potential.

In theory, iris scanning has an equal error rate of about $10^{-5}$. But to achieve such spectacular results, the enrollment phase must be extremely accurate. If the real-world enrollment environment is not up to laboratory standards, then the results might not be so impressive.

Undoubtedly many inexpensive biometrics systems fare worse than the results given in Table 7.2. And biometrics in general have a very poor record with respect to the inherently difficult identification problem.

### 7.4.4 Biometric Conclusions

Biometrics clearly have many potential advantages over passwords. In particular, biometrics are difficult, although not impossible, to forge. In the case of fingerprints, Trudy could steal Alice's thumb, or, in a less gruesome attack, Trudy might be able to use a copy of Alice's fingerprint. Of course, a more sophisticated system might be able to detect such an attack, but then the system will be more costly, thereby reducing its desirability as a replacement for passwords.[12]

[12]Unfortunately for security, passwords are likely to remain free for the foreseeable future.

There are also many potential software-based attacks on authentication. For example, it may be possible to subvert the software that does the comparison or to manipulate the database that contains the enrollment data. Such attacks apply to most authentication systems, regardless of whether they are based on biometrics, passwords, or other techniques.

While a broken cryptographic key or password can be revoked and replaced, it's not clear how to revoke a broken biometric. This and other biometric pitfalls are discussed by Schneier [260].

Biometrics have a great deal of potential as a substitute for passwords, but biometrics are not foolproof. And given the enormous problems with passwords and the vast potential of biometrics, it's perhaps surprising that biometrics are not more widely used today. This should change in the future as biometrics become more robust and inexpensive.

## 7.5 Something You Have

Smartcards or other hardware tokens can be used for authentication. Such authentication is based on the "something you have" principle. A smartcard is a credit card sized device that includes a small amount of memory and computing resources, so that it is able to store cryptographic keys or other secrets, and perhaps even do some computations on the card. A special-purpose smartcard reader, as shown in Figure 7.7, is used to read the key stored on the card. Then the key can be used to authenticate the user. Since a key is used, and keys are selected at random, password guessing attacks can be eliminated.[13]

Figure 7.7: A Smartcard Reader (Courtesy of Athena, Inc.)

There are several other examples of authentication based on "something you have," including a laptop computer (or its MAC address), an ATM card, or a password generator. Here, we give an example of a password generator.

[13]Well, a PIN might be required to access the key, so password issues might still arise.

A password generator is a small device that the user must have (and use) to log in to a system. Suppose that Alice has a password generator, and she wants to authenticate herself to Bob. Bob sends a random "challenge" R to Alice, which Alice then inputs into the password generator along with her PIN number. The password generator then produces a response, which Alice transmits to Bob. If the response is correct, Bob is convinced that he's indeed talking to Alice, since only Alice is supposed to have the password generator. This process is illustrated in Figure 7.8.

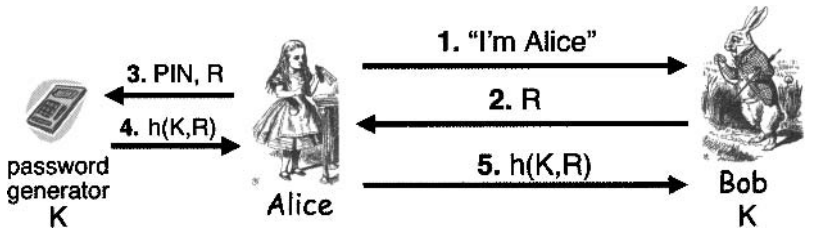


Figure 7.8: Password Generator

For a challenge-response authentication scheme to work, Bob must be able to verify that Alice's response is correct. For the example in Figure 7.8, Bob and the password generator must both have access to the key $K$, since the password generator needs the key to compute the hash, and Bob needs the key to verify Alice's response. Alice accesses the key $K$ only indirectly—by entering her PIN into the key generator. We'll see more examples of the use of challenge-response mechanisms in the upcoming chapters on security protocols.

## 7.6 Two-Factor Authentication

In fact, the password generator scheme in Figure 7.8 requires both "something you have" (the password generator) and "something you know" (the PIN). Any authentication method that requires two out of the three "somethings" is known as *two-factor authentication.* Another example of a two-factor authentication is an ATM card, where the user must have the card and know the PIN number. Other examples of two-factor authentication include a credit card together with a signature, a biometric thumbprint system that also requires a password, and a cell phone that requires a PIN.

## 7.7 Single Sign-On and Web Cookies

Before concluding this chapter, we briefly mention two additional authentication topics. First, we discuss *single sign-on*, which is a topic of considerable

practical importance. We'll also briefly mention *Web cookies*, which are often used as a weak form of authentication.

Users find it troublesome to enter their authentication information (typically, passwords) repeatedly. For example, when browsing the Web, it is not uncommon that many different websites require passwords. While this is sensible from a security perspective, it places a burden on users who must either remember different passwords for many different websites or compromise their security by reusing passwords.

A more convenient solution would be to have Alice authenticate once and then have a successful result automatically "follow" her wherever she goes on the Internet. That is, the initial authentication would require Alice's participation, but subsequent authentications would happen behind the scenes. This is known as single sign-on, and single sign-on for the Internet has been a topic of some interest for several years.

As with many computing topics, there are competing and incompatible approaches to single sign-on for the Internet. As is often the case, there is the Microsoft way, and the "everybody else" way. The approach favored by Microsoft goes by the name of Passport [171, 203], while the method preferred by (nearly) everybody else is the Liberty Alliance [100, 192]. The latter approach is based on the Security Assertion Markup Language, or SAML [78].

Certainly, a secure single sign-on for the Internet would be a major convenience. However, it does not appear that any such method is likely to gain widespread acceptance any time soon. It is worth noting that we will see a single sign-on architecture in Chapter 10 when we discuss the Kerberos security protocol.

Finally, we mention Web cookies, which have some interesting security implications. When Alice is surfing the Web, websites often provide Alice's browser with a Web cookie, which is simply a numerical value that is stored and managed by Alice's browser. The website also stores the cookie, which is used to index a database that retains information about Alice.

When Alice returns to a website for which she has a cookie, the cookie is automatically passed by her browser to the website. The website can then access its database to remember important information about Alice. In this way, cookies maintain state across sessions. Since the Web uses HTTP, which is a stateless protocol, cookies are also used to maintain state within a session.

In a sense, cookies can act as a single sign-on method for a website. That is, a website can authenticate "Alice" based on the possession of Alice's Web cookie. Or, in a slightly stronger version, a password is used to initially authenticate Alice, after which the cookie is considered sufficient. Either way, this is a fairly weak form of authentication, but it illustrates the often irresistible temptation to use whatever is available and convenient as a security mechanism, whether it is secure or not.

## 7.8 Summary

You can authenticate to a machine based on "something you know," "something you have," or "something you are." Passwords are synonymous with the "something you know" method of authentication. In this chapter, we discussed passwords at length. The bottom line is that passwords are far from an ideal method of authentication, but they are likely to remain popular for the foreseeable future, since passwords are the lowest cost option.

We also discussed authentication based on "something you are," i.e., biometrics. It is clear that biometrics offer the potential for much higher security than passwords. However, biometrics cost money, and they are not entirely without problems.

We briefly mentioned "something you have" methods of authentication, as well as two-factor authentication, which combines any two of the three methods. Finally, we briefly discussed single sign-on and Web cookies.

In the next chapter, we'll discuss authorization, which deals with restrictions placed on authenticated users. The authentication problem returns to the fore in Chapters 9 and 10, where we cover security protocols. We'll see that authentication over a network is a whole nother can of worms.

## 7.9 Problems

1. As discussed in this chapter, relatively strong passwords can be derived from passphrases.

   a. Give two passwords derived from the passphrase "Gentlemen do not read other gentlemen's mail."

   b. Give two passwords derived from the passphrase "Are you who you say you are?"

2. For each of the following passwords, give a passphrase that the password could have been derived from.

   a. `PokeGCTall`

   b. `4s&7yrsa`

   c. `gimmeliborD`

   d. `IcntgetNOsat`

3. In the context of biometrics, define the terms *fraud rate* and *insult rate.* In statistics, which of these is a Type I error and which is a Type II error?

4. In some applications, a passcode consisting of some number of digits is required (for example, a PIN). Using the number-to-letter conversion on a telephone,

   a. What passcode corresponds to the password "hello"?

   b. Find as many passwords as you can that correspond to the passcode 5465, where each password is an English dictionary word.

5. Suppose that on a particular system, all passwords are 10 characters, there are 64 choices for each character, and the system has a password file containing 512 hashed passwords. Furthermore, Trudy has a dictionary of $2^{20}$ common passwords. Provide pseudo-code for an efficient attack on the password file in the following cases.

   a. The password hashes are not salted.

   b. The password hashes are salted.

6. This problem deals with storing passwords in a file.

   a. Why is it a good idea to hash passwords that are stored in a file?

   b. Why is it a much better idea to hash passwords stored in a file than to encrypt the password file?

   c. What is a salt and why should a salt be used whenever passwords are hashed?

7. On a particular system, all passwords are 8 characters, there are 128 choices for each character, and there is a password file containing the hashes of $2^{10}$ passwords. Trudy has a dictionary of $2^{30}$ passwords, and the probability that a randomly selected password is in her dictionary is 1/4. Work is measured in terms of the number of hashes computed.

   a. Suppose that Trudy wants to recover Alice's password. Using her dictionary, what is the expected work for Trudy to crack Alice's password, assuming the passwords are not salted?

   b. Repeat part a, assuming the passwords are salted.

   c. What is the probability that at least one of the passwords in the password file appears in Trudy's dictionary?

8. Suppose you are a merchant and you decide to use a biometric fingerprint device to authenticate people who make credit card purchases at your store. You can choose between two different systems: System A has a fraud rate of 1% and an insult rate of 5%, while System B has a fraud rate of 5% and an insult rate of 1%.

   a. Which system is more secure and why?

b. Which system is more user-friendly and why?

c. Which system would you choose and why?

9. Research has shown that most people cannot accurately identify an individual from a photo. For example, one study found that most people will accept an ID with any photo that has a picture of a person of the same gender and race as the presenter.

   a. It has also been demonstrated that when photos are included on credit cards, the fraud rate drops significantly. Explain this apparent contradiction.

   b. Your easily amused author frequents an amusement park that provides each season passholder with a plastic card similar to a credit card. The park takes a photo of each season passholder, but the photo does not appear on the card. Instead, when the card is presented for admission to the park, the photo appears on a screen that is visible to the park attendant. Why might this approach be better than putting the photo on the card?

10. Suppose all passwords on a given system are 8 characters and that each character can be any one of 64 different values. The passwords are hashed (with a salt) and stored in a password file. Now suppose Trudy has a password cracking program that can test 64 passwords per second. Trudy also has a dictionary of $2^{30}$ common passwords and the probability that any given password is in her dictionary is $1/4$. The password file on this system contains 256 password hashes.

    a. How many different passwords are possible?

    b. How long, on average, will it take Trudy to crack the administrator's password?

    c. What is the probability that at least one of the 256 passwords in the password file is in the dictionary?

    d. What is the expected work for Trudy to recover any one of the passwords in the password file?

11. Let $h$ be a secure cryptographic hash function. For this problem, a password consists of a maximum of 14-characters and there are 32 possible choices for each character. If a password is less than 14 characters, it's padded with nulls until it is exactly 14 characters. Let $P$ be the resulting 14 character password. Consider the following two password hashing schemes.

    (i) The password $P$ is split into two parts, with $X$ equal to the first 7 characters and $Y$ equal to the last 7 characters. The password is stored as $(h(X), h(Y))$. No salt is used.

(ii) The password is stored as $h(P)$. Again, no salt is used.

Note that the method in scheme (i) is used in Windows to store the so-called LANMAN password.

a. Assuming a brute force attack, how much easier is it to crack the password if scheme (i) is used as compared with scheme (ii)?

b. If scheme (i) is used, why might a 10-character password be *less* secure than a 7-character password?[14]

12. Suppose that passwords are stored as follows, where there are 128 possible choices for each character: If a password exceeds 16 characters, it is truncated to 16 characters. If a password is less than 16 characters, it is padded with "A" until it is exactly 16 characters. The resulting 16-character password is split into two parts, $X_0$ and $X_1$, where $X_0$ consists of the first six characters and $X_1$ consists of the last 10 characters. The password is hashed as $Y_0 = h(X_0, S_0)$ and $Y_1 = h(X_1, S_1)$, where $S_0$ and $S_1$ are each 64-bit salt values. The values $(Y_0, S_0)$ and $(Y_1, S_1)$ are stored for use in password verification.

a. Precisely how are $(Y_0, S_0)$ and $(Y_1, S_1)$ used to verify an entered password?

b. What is the expected work for an exhaustive search to recover one particular password (for example, the administrator's password)?

c. How would you attack a password in a way that could provide a significant shortcut over an exhaustive search or a standard dictionary attack? Explain.

13. Many websites require users to register before they can access information or services. Suppose that you register at such a website, but when you return later you've forgotten your password. The website then asks you to enter your email address, which you do. Later, you receive your original password via email.

a. Discuss several security concerns with this approach to dealing with forgotten passwords.

b. The correct way to deal with passwords is to store salted hashes of passwords. Does this website use the correct approach? Justify your answer.

[14]In fact, the standard advice for LANMAN passwords is that users should choose either a 7-character password, or a 14-character password, since anything in between these two lengths is less secure.

14. Alice forgets her password. She goes to the system administrator's office, and the admin resets her password and gives Alice the new password.

    a. Why does the SA reset the password instead of giving Alice her previous (forgotten) password?

    b. Why should Alice re-reset her password immediately after the SA has reset it?

    c. Suppose that after the SA resets Alice's password, she remembers her previous password. Alice likes her old password, so she resets it to its previous value. Would it be possible for the SA to determine that Alice has chosen the same password as before? Why or why not?

15. Consider the password generator in Figure 7.8.

    a. If $R$ is repeated, is the protocol secure?

    b. If $R$ is predictable, is the protocol secure?

16. Describe attacks on an authentication scheme based on Web cookies.

17. Briefly outline the most significant technical differences between Passport and Liberty Alliance.

18. MAC address are globally unique and they don't change except in rare instances where hardware changes.

    a. Explain how the MAC address on your computer could be used as a "something you have" form of authentication.

    b. How could you use the MAC address as part of a two-factor authentication scheme?

    c. How secure is your authentication scheme in part a? How much more secure is your authentication scheme in part b?

19. Suppose you have six accounts, each of which requires a password, and you choose distinct passwords for each account.

    a. If the probability that any given password is in Trudy's password dictionary is 1/4, what is the probability that at least one of your passwords is in Trudy's dictionary?

    b. If the probability that any one of your passwords is in Trudy's dictionary is reduced to 1/10, what is the probability that at least one of your passwords is in Trudy's dictionary?

20. Suppose that you have $n$ accounts, each of which requires a password. Trudy has a dictionary and the probability that a password appears in Trudy's dictionary is $p$.

    a. If you use the same password for all $n$ accounts, what is the probability that your password appears in Trudy's dictionary?

    b. If you use distinct passwords for each of your $n$ accounts, what is the probability that at least one of your passwords appears in Trudy's dictionary? Show that if $n = 1$, your answer agrees with your answer to part a.

    c. Which is more secure, choosing the same password for all accounts, or choosing different passwords for each account? Why? See also Problem 21.

21. Suppose that Alice uses two distinct passwords—one strong password for sites where she believes security is important (e.g., her online bank), and one weak password for sites where she does not care much about security (e.g., social networking sites).

    a. Alice believes this is a reasonable compromise between security and convenience. What do you think?

    b. What are some practical difficulties that might arise with such an approach?

22. Suppose Alice requires passwords for eight different accounts. She could choose the same password for all of these accounts. With just a single password to remember, Alice might be more likely to choose a strong password. On the other hand, Alice could choose different passwords for each account. With distinct passwords, she might be tempted to choose weaker passwords since this might make it easier for her to remember all of her passwords.

    a. What are the trade-offs between one well-chosen password versus several weaker passwords?

    b. Is there a third approach that is more secure than either of these options?

23. Consider Case I from Section 7.3.5.

    a. If the passwords are unsalted, how much work is it for Trudy to precompute all possible hash values?

    b. If each password is salted with a 16-bit value, how much work is it for Trudy to precompute all possible hash values?

c. If each password is salted with a 64-bit value, how much work is it for Trudy to precompute all possible hash values?

24. Suppose that Trudy has a dictionary of $2^n$ passwords and the probability that a given password is in her dictionary is $p$. If Trudy obtains a file containing a large number of salted password hashes, show that the expected work to recover a password is bounded by $2^{n-1}(1+2(1-p)/p)$. Hint: As in Section 7.3.5, Case IV, ignore the highly improbable case where none of the passwords in the file appears in Trudy's dictionary. Then make use of the fact that $\sum_{k=0}^{\infty} x^k = 1/(1-x)$ and also $\sum_{k=1}^{\infty} kx^k = x/(1-x)^2$, provided $|x| < 1$.

25. For password cracking, generally the most realistic situation is Case IV of Section 7.3.5. In this case, the amount of work that Trudy must do to determine a password depends on the size of the dictionary, the probability that a given password is in the dictionary, and the size of the password file. Suppose Trudy's dictionary is of size $2^n$, the probability that a password is in the dictionary is $p$, and the password file is of size $M$. Show that if $p$ is small and $M$ is sufficiently large, then Trudy's expected work is about $2^n/p$. Hint: Use the result of Problem 24.

26. Suppose that when a fingerprint is compared with one other (non-matching) fingerprint, the chance of a false match is 1 in $10^{10}$, which is approximately the error rate when 16 points are required to determine a match (the British legal standard). Suppose that the FBI fingerprint database contains $10^7$ fingerprints.

    a. How many false matches will occur when 100,000 suspect fingerprints are each compared with the entire database?

    b. For any individual suspect, what is the chance of a false match?

27. Suppose DNA matching could be done in real time.

    a. Describe a biometric for secure entry into a restricted facility based on this technique.

    b. Discuss one security concern and one privacy concern with your proposed system in part a.

28. This problem deals with biometrics.

    a. What is the difference between the authentication problem and the identification problem?

    b. Which is the inherently easier problem, authentication or identification? Why?

29. This problem deals with biometrics.

    a. Define fraud rate.

    b. Define insult rate.

    c. What is the equal error rate, how is it determined, and why is it useful?

30. Gait recognition is a biometric that distinguishes based on the way a person walks, whereas a digital doggie is a biometric that distinguishes based on odor.

    a. Describe an attack on gait recognition when it's used for identification.

    b. Describe an attack on a digital doggie when it's used for identification.

31. Recently, facial recognition has been touted as a possible method for, say, identifying terrorists in airports. As mentioned in the text, facial recognition is used by Las Vegas casinos in an attempt to detect cheaters. Note that in both of these cases the biometric is being used for identification (not authentication), presumably with uncooperative subjects.

    a. Discuss an attack on facial recognition when used by a casino to detect cheaters.

    b. Discuss a countermeasure that casinos might employ to reduce the effectiveness of your attack in part a.

    c. Discuss a counter-countermeasure that attackers might employ to reduce the effectiveness of your countermeasure in b.

32. In one episode of the television show *MythBusters*, three successful attacks on fingerprint biometrics are demonstrated [213].

    a. Briefly discuss each of these attacks.

    b. Discuss possible countermeasures for each of the attacks in part a. That is, discuss ways that the biometric systems could be made more robust against the specific attacks.

33. This problem deals with possible attacks on a hand geometry biometric system.

    a. Discuss analogous attacks to those in Problem 32 but for a hand geometry biometric system.

b. In your judgment, which would be more difficult to break, the fingerprint door lock in Problem 32, or an analogous system based on hand geometry? Justify your answer.

34. A retina scan is an example of a well-known biometric that was not discussed in this chapter.

    a. Briefly outline the history and development of the retina scan biometric. How does a modern retina scan system work?

    b. Why, in principle, can a retina scan be extremely effective?

    c. List several pros and cons of retina scanning as compared to a fingerprint biometric.

    d. Suppose that your company is considering installing a biometric system that every employee will use every time they enter their office building. Your company will install either a retina scan or an iris scan system. Which would you prefer that they choose? Why?

35. A sonogram is a visual representation of sound. Obtain and install a speech analysis tool that can generate sonograms.[15]

    a. Examine several sonograms of your voice, each time saying "open sesame." Qualitatively, how similar are the sonograms?

    b. Examine several sonograms of someone else saying "open sesame." How similar are these sonograms to each other?

    c. In what ways do your sonograms from part a differ from those in part b?

    d. How would you go about trying to develop a reliable biometric based on voice recognition? What characteristics of the sonograms might be useful for distinguishing speakers?

36. This problem deals with possible attacks on an iris scan biometric system.

    a. Discuss analogous attacks to those in Problem 32 on an iris scan biometric system.

    b. Why would it be significantly more difficult to break an iris scan system than the fingerprint door lock in Problem 32?

    c. Given that an iris scan biometric is inherently stronger than a fingerprint-based biometric system, why are fingerprint biometrics far more popular?

[15]Your audacious author uses Audacity [20] to record speech and Sonogram [272] to generate sonograms and analyze the resulting audio files. Both of these are freeware.

37. Suppose that a particular iris scan systems generates 64-bit iris codes instead of the standard 2048-bit iris codes mentioned in this chapter. During the enrollment phase, the following iris codes (in hex) are determined.

| User | Iris code |
|---|---|
| Alice | BE439AD598EF5147 |
| Bob | 9C8B7A1425369584 |
| Charlie | 885522336699CCBB |

During the recognition phase, the following iris codes are obtained.

| User | Iris code |
|---|---|
| U | C975A2132E89CEAF |
| V | DB9A8675342FEC15 |
| W | A6039AD5F8CFD965 |
| X | 1DCA7A54273497CC |
| Y | AF8B6C7D5E3F0F9A |

Use the iris codes above to answer the following questions.

a. Use the formula in equation (7.1) to compute the following distances:

$$d(\text{Alice}, \text{Bob}),\ d(\text{Alice}, \text{Charlie}),\ d(\text{Bob}, \text{Charlie}).$$

b. Assuming that the same statistics apply to these iris codes as the iris codes discussed in Section 7.4.2.3, which of the users, U,V,W,X,Y, is most likely Alice? Bob? Charlie? None of the above?

38. A popular "something you have" method of authentication is the RSA SecurID [252]. The SecureID system is often deployed as a USB key. The algorithm used by SecurID is similar to that given for the password generator illustrated in Figure 7.8. However, no challenge $R$ is sent from Bob to Alice; instead, the current time $T$ (typically, to a resolution of one minute) is used. That is, Alice's password generator computes $h(K, T)$ and this is sent directly to Bob, provided Alice has entered the correct PIN (or password).

a. Draw a diagram analogous to that in Figure 7.8 illustrating the SecurID algorithm.

b. Why do we need $T$? That is, why is the protocol insecure if we remove $T$?

c. What are the advantages and disadvantages of using the time $T$ as compared to using a random challenge $R$?

d. Which is more secure, using a random challenge $R$ or the time $T$? Why?

39. A password generator is illustrated in Figure 7.8.

a. Discuss possible cryptanalytic attacks on the password generator scheme in Figure 7.8.

b. Discuss network-based attacks on the password generator scheme in Figure 7.8.

c. Discuss possible non-technical attacks on the password generator scheme in Figure 7.8.

40. In addition to the holy trinity of "somethings" discussed in this chapter (something you know, are, or have), it is also possible to base authentication on "something you do." For example, you might need to press a button on your wireless access point to reset it, proving that you have physical access to the device.

a. Give another real-world example where authentication could be based on "something you do."

b. Give an example of two-factor authentication that includes "something you do" as one of the factors.