EBOOK EXTRAS: v4.2
Downloads, Updates, Feedback



TAKE CONTROL OF

Your Passwords

by JOE KISSELL

$14.99

4TH EDITION

Table of Contents

Read Me First 5
- Updates and More 5
- Basics 6
- What's New in Version 4.2 6
- What Was New in Version 4.1.1 6
- What Was New in Version 4.1 7
- What Was New in the Fourth Edition 7

Introduction 9

Passwords Quick Start 12

Understand the Problems with Passwords 13
- Simple for You, Simple for Them 13
- The One and the Many 14
- The Major Threats 15
- Timeworn Tricks 25
- Usernames and Passwords: an Outdated Model 27

Learn About Password Security 35
- What Makes a Good Password? 35
- All About Entropy 36
- Why a Great Password Isn't Enough 42
- What About Mobile Device Passcodes? 43
- Understanding Security Questions and Reset Procedures 44
- Multi-Factor Authentication 47
- Authenticating with Another Site's Credentials 56

Apply Joe's Password Strategy 59
- Figure Out Which Passwords You Must Memorize 60
- Create Strong but Memorable Passwords 61
- Use a Password Manager for Everything Else 65
- Handle Security Questions 72
- Manage Email Options 73
- Deal with Exceptions and Surprises 74

Pick a Password Manager 79
Features to Look For 80
Example Password Managers 87
Joe's Recommendations 108
Keep Your Passwords Secure 109
Avoid the "Weakest Link" Problem 109
Back Up Your Passwords 111
Prepare an Emergency Password Plan 112
Audit Your Passwords 116
Understand the Overall Process 117
Look for Weak Passwords 118
Triage Your Passwords 119
Update a Password 120
Check for Compromised and Vulnerable Passwords 124
Authenticate Without Passwords 126
How Passkeys Work 127
Operating System & Browser Integration 129
How to Create and Use a Passkey 131
Syncing Passkeys Across Devices 132
Appendix A: Use Two-Factor Authentication 135
Two-Step Verification Basics 135
Use Apple's Enhanced Security Options 137
Use Dropbox's Two-Step Verification 145
Use Facebook's Two-Step Verification 146
Use Google's Two-Step Verification 147
Use Microsoft's Two-Step Verification 148
Appendix B: Help Your Uncle with His Passwords 149
Password Manager Compromises 149
Password Reuse Compromises 150
Password Complexity Compromises 151
Appendix C: Calculate Password Strength 152
An Aside: Doing Math with Google 152
The Entropy Formula 154

Why That Entropy Formula Is Wrong 157
Back to zxcvbn 159
Password Strength Summary 160
For Further Reading 160
About This Book 161
Ebook Extras 161
About the Author and Publisher 162
Credits 162
Also by Joe Kissell 163
Copyright and Fine Print 164

Read Me First

Welcome to *Take Control of Your Passwords, Fourth Edition,* version 4.2, published in January 2025 by alt concepts. This book was written by Joe Kissell and edited by Kelly Turner.

Passwords are an irritating fact of modern life. It's tricky to create and remember good ones, but dangerous to use simple ones (or reuse a password in multiple places). This book helps you overcome these problems with a sensible, stress-free strategy for password security.

If you want to share this ebook with a friend, we ask that you do so as you would with a physical book: "lend" it for a quick look, but ask your friend to buy a copy for careful reading or reference. Discounted classroom and user group copies are available.

Copyright © 2025, Joe Kissell. All rights reserved.

Updates and More

You can access extras related to this ebook on the web (use the link in Ebook Extras, near the end; it's available only to purchasers). On the ebook's Take Control Extras page, you can:

- Download any available new version of the ebook for free, or buy any subsequent edition at a discount.
- Access the book in both PDF and EPUB formats. (Learn about reading on mobile devices on our Device Advice page.)
- Read the ebook's blog. You may find new tips or information, as well as a link to an author interview.

If you bought this ebook from the Take Control website, it has been added to your account, where you can download it in other formats and access any future updates.

Basics

Be aware of the following:

- **Credentials:** I frequently use the term "credentials" as a compact way of saying "the combination of your username and password." In some cases, additional pieces of information, such as your ZIP code or the answers to security questions, may be considered part of your credentials—it's whatever a site or service needs to reliably identify you as the authorized user of a given account.
- **Authentication:** The act of proving your identity to a computer system—typically by entering your credentials and having them confirmed as matching the previously stored record—is called *authentication*. I use that term a number of times in this book, so I want to make sure you're familiar with it.

What's New in Version 4.2

Version 4.2 covers changes in the industry, and especially at Apple, since the previous version of this book was published. Most notably:

- In Threat #3: Brute-Force Attacks, added information about how encryption algorithms affect cracking time
- Revised Apple's Password Tools (previously titled "iCloud Keychain") to cover the Passwords app introduced in macOS 15 Sequoia, iOS 18, and iPadOS 18 and provide more information on the iCloud Passwords software
- Made numerous updates to the Authenticate Without Passwords chapter to reflect new passkey-related developments (and ongoing limitations)

What Was New in Version 4.1.1

This update reflected the fact that Proton Pass now supports passkeys.

What Was New in Version 4.1

Version 4.1 updated the book to cover changes in operating systems, browsers, and third-party password managers since mid-2023. The biggest theme in all those areas was increased support for passkeys. Major changes in this book included:

- Added passkey support to the Features to Look For list
- Updated Example Password Managers with the current details for each password manager, and added a new one: Proton Pass
- Revised Operating System & Browser Integration and Syncing Passkeys Across Devices with more up-to-date details, especially about passkey handling
- Removed information on Twitter's two-factor authentication from Appendix A: Use Two-Factor Authentication

What Was New in the Fourth Edition

Since the previous edition of this book, the topic of password security had grown considerably more complex. Version 4.0 updated the book to cover the latest technologies, including the following significant changes:

- Updated Usernames and Passwords: an Outdated Model with information on passkeys, magic links (see the sidebar What About Magic Links?), and software changes.
- Mentioned Sign in with Apple and Hide My Email in Should Usernames Be Unique and Random Too?.
- Added a new topic that addresses the normally short, numeric passcodes used to unlock phones and tablets; see What About Mobile Device Passcodes?.
- In Physical Keys, I now discuss newer security key variants.

- Thoroughly updated Example Password Managers to cover the latest app versions, capabilities, and prices. I also updated the Missing Managers sidebar to talk about why some password managers no longer appear in the book, and I added an entirely new sidebar to more fully explain the situation with LastPass; see The Decline and Fall of LastPass.
- Removed the topic "Use Wireless Networks Safely," which was too tangential to the topic of the book, and didn't accurately reflect modern security norms.
- Added a big new chapter, Authenticate Without Passwords, that provides detailed information on passkeys—the up-and-coming technology that may eventually replace passwords for good.

Introduction

Think of a card, any card. Now, keep that card in mind and think of another. Repeat until you've picked 16 cards—but make sure your selection includes all four suits, at least one ace and one face card, and no two instances of the same card. Remember the whole set, because I'm going to ask you to name them all tomorrow…

I'm joking, of course. But have you ever noticed that when magicians pull someone out of an audience to help with a trick, they never make such complicated requests? It's unreasonable to ask someone to create a meaningless string of numbers and letters, remember it indefinitely, and produce it on demand.

But websites, banks, and network administrators make exactly that request of us almost daily. Want to buy something online? Sure, but you need more than a credit card—you usually need a password too. Sync this data with the cloud, sign up for that free service, manage your utilities or PTA schedule online…no problem, but you must have a password for that. "Make sure it's between 12 and 16 characters, contains upper- and lowercase letters, at least one digit, at least one punctuation character (but not *that* punctuation character!), and doesn't have any repeated strings. Oh yeah, and don't even think about using a word that might be found in a dictionary or reusing a password you used anywhere else."

Are you kidding me? This is madness. Coming up with unique, random passwords all the time, remembering them, and producing them reliably is not the sort of task the human brain is cut out for.

Faced with this difficult and increasingly absurd task, people naturally tend to look for shortcuts their brains can handle. They pick easy passwords, like their kids' names or patterns of keys on the keyboard. Even if they go to the effort of creating something more complex, they use the same password everywhere, because then they have only one thing to remember instead of hundreds.

Speaking as a fellow human being, I don't blame anyone for taking the easy way out. You might try to come up with clever, random-looking passwords the first few times, but once your list of password-protected accounts grows into the dozens, and then the hundreds, it's not plausible to keep following the rules.

However, speaking as a technologist who has spent lots of time researching and thinking about security, I'm terrified for people who do this. I know how easy it is to guess, crack, or otherwise uncover someone's passwords, because I've done it myself. And people with far greater skills and resources than mine spend all day, every day doing the same thing—not for legitimate security research but to steal money and secrets, to cause mischief, or to show off.

Every few weeks I read about another high-profile case in which millions of passwords are leaked, hacked, or stolen. And then I look at that list of now-public passwords and shake my head when I see that thousands of folks thought `password` was a pretty good password! I understand why they did it—they were only trying to manage an unmanageable problem—but I feel sorry for them, as their problems didn't end with the site that was hacked. Because these people invariably use the same password on lots of sites, many of them had money and identities stolen, private email messages read, or hate mail sent in their name. It's a big, scary deal.

Methods of managing passwords that might have worked well a decade or two ago (including some that I recommended myself) could be downright dangerous today, as hackers and their tools have become much more sophisticated, and the stakes have gone up considerably. On the bright side, the apps and techniques available to us good guys have improved too. While I can't solve all the world's password problems, with a combination of technology and common sense, I can probably help you solve about 98% of *your* password problems.

My goal in this book is to lay out a simple strategy that will keep you as secure as possible with a minimum of effort. Sometimes, I admit, there's a trade-off between security and convenience. You have to choose which is more upsetting: adding another lock to your door or risking a break-in because the neighborhood's gotten worse. But you

might be surprised to discover that in many cases, you can significantly increase your security without extra effort. Remember how I said that generating and remembering random passwords is not something the human brain is good at? That's true, but I'll bet every human reading this book has a smartphone as well as a tablet, desktop, or laptop computer, and those devices are *fantastic* at generating and remembering passwords—if you use the right apps, in the right ways, at the right times. (And yes, I'll also talk about the situations in which your gadgets can't help you. Don't worry; those problems have solutions too.)

If all this talk of hacking and identity theft sounds scary, I'm sorry. I don't mean to frighten you. Much. But I do want you to have a clear understanding of the threats so you're motivated to adopt better password practices. It won't take long, it won't cost much, and it won't be difficult. Once you've done it, you can go back to not being scared, just like me. In fact, that's the point of my recommendations—I want you to be relaxed and confident, knowing that your passwords are solid and that you have an easy, reliable way to create and enter passwords whenever they're needed.

In this book, I look at the problem of passwords in a broad, platform-agnostic way. Whether you use a Mac or PC, an iPhone, an iPad, or an Android device, something else entirely, or—more likely—a combination, you'll find guidance to help you take control of your passwords. By the end of this book, I hope you'll thoroughly understand the vulnerabilities and threats associated with passwords, ways to minimize your risks, and how to use passwords safely without losing your sanity. You'll also learn how you can dip your toes in the passwordless future that technology companies have been promising us for decades.

No one can give you an ironclad promise of perfect, unbreakable security, but with the advice in this book, I can get you pretty darn close.

Passwords Quick Start

I recommend reading this book in linear order, because each chapter builds on what comes before it. In any case, don't skip Apply Joe's Password Strategy, because using just part of my strategy (such as a password manager) may leave important gaps in your security.

Get your bearings:

- Find out what's wrong with passwords and the ways most people use them; see Understand the Problems with Passwords.
- Discover what makes a good password and why that's not all you have to worry about; see Learn About Password Security.

Develop your password toolkit:

- Learn my three-point password strategy—and what to do in situations that don't fit into it; see Apply Joe's Password Strategy.
- Arm yourself with a good app for creating, remembering, and entering random passwords; see Pick a Password Manager.

Tie up loose ends and fix old problems:

- Make sure your passwords don't fall into the wrong hands while remaining available when needed; see Keep Your Passwords Secure.
- Clean up all those awful passwords you created before you saw the light; see Audit Your Passwords.

Handle special cases:

- Use passkeys when you can; see Authenticate Without Passwords.
- Deal with systems that use a password plus another authentication method; see Appendix A: Use Two-Factor Authentication.
- Get advice for improving password security for someone who's unwilling or unable to follow my regular strategy; see Appendix B: Help Your Uncle with His Passwords.
- Learn the math behind password entropy; see Appendix C: Calculate Password Strength.

Understand the Problems with Passwords

Because you're reading this book, you probably already have a problem with your passwords, such as how to come up with them or how to remember them. We'll get to those sorts of problems shortly.

First, I want to discuss some of the overall problems with passwords. What's wrong with simple, easy-to-remember passwords? Why do we need so many passwords, anyway? What are the common threats against passwords? And if this whole username/password system is so flawed, what can be done about it?

Simple for You, Simple for Them

The whole idea of a password is that it's private—something known only to you and to the entity with which you have an account (a bank, website, cloud service, etc.). If someone else learns your password, that person can access your data, and that's just the beginning.

Once access is granted, the interloper—I'll refer to this hypothetical person as a "hacker" even though that's not necessarily accurate—can change your password so you can't access your own account, impersonate you online, and even change your contact data to theirs. And, if you use the same password for other sites and services, the hacker can get access just as easily to your other accounts and wreak all kinds of havoc, up to and including "stealing" your identity (and your money).

Obviously, I'm talking about a worst-case scenario. Most password breaches result in less-serious problems—comparable to someone picking the lock to your house, but not actually taking anything of value. Even so, I think most of us would prefer to avoid that icky feeling that a stranger has been poking around in our personal space, and hassles like changing the locks.

So, your goal when selecting any new password should be to reduce, as far as possible, the likelihood that someone else can discover what it is. You essentially want locks that are strong enough to ward off those unlikely worst-case scenarios, thereby protecting yourself against less-serious risks in the process.

You might be surprised at the ways in which someone could discover your password; I talk about many of these in the remainder of this chapter. But let me start with what I hope is obvious by now: The passwords that are the simplest for you to use are also the simplest for a hacker to discover. Those are the passwords to avoid at all costs.

When someone says that you should never pick a password that's a word in a dictionary, the name of a relative or pet, the date of your anniversary, or another easy-to-remember string, they're pointing out the insecurity of highly guessable passwords. If I wanted to break into an account belonging to someone I knew (a coworker, say), I'd certainly try as many terms like these as I could think of, hoping that what's easy for them to remember is also easy for me to guess.

Of course, you're not merely up against flesh-and-blood guessers. Computers can do an even better and faster job of guessing passwords. You need passwords that are unguessable by human *or* machine. Such passwords are often, unfortunately, hard to remember and type too, which is why they aren't used more often. As this book progresses, I'll explain my suggested strategy for dealing with this problem. For now, remember: a simple password is nearly as bad as no password at all.

The One and the Many

One of the recurring themes in this book—I want to repeat it until you believe—is that reusing passwords is a terrible, terrible idea. Just. Don't. Ever. Do. It.

The basic argument is simple. If your password for one site or service is compromised (stolen, leaked, guessed, hacked) and you also used that password somewhere else, then whoever has your password might try it elsewhere and be able to do that much more damage. If you use

the same password everywhere, you're essentially handing over all your personal data and access to the first person who learns your password.

Even if you make every effort to keep your password safe, you can't count on every provider where you use that password doing the same. Websites are hacked and passwords stolen or leaked all the time. Because you can't trust every organization that has your password to protect it, your best and safest defense is never to use the same password twice. This doesn't have to be hard at all, honest. (I explain how you can pull it off without going crazy in Apply Joe's Password Strategy, a couple of chapters ahead.)

Despite having preached against reusing passwords for years, I still encounter people who insist they *just can't* be bothered to come up with new ones all the time and *really couldn't* remember more than a handful of them anyway. And you know what? There are a few topics—religion, politics, computing platforms, and reusing passwords—that are futile to argue about. I'm sure that *you* are sensible enough to have seen the light and you wouldn't even consider reusing passwords. But what about your cranky uncle who's more set in his ways? Don't sweat it. Read Appendix B: Help Your Uncle with His Passwords for some compromises that still leave your uncle reasonably secure.

The Major Threats

Someone has asked you to create a password, and you've done it. At that moment, only you and the entity (person, software, website, whatever) on the other end know your password, so it *should* be entirely secure, right? What could possibly go wrong?

Well, lots of things can and do go wrong. To help you understand what you're facing and create a smart strategy to deal with it, allow me to mention five of the most significant threats to password security. As you'll see, some of these threats are highly unlikely to affect the average person, but the steps that protect you against the more prevalent threats will also protect you against the more obscure ones.

Threat #1: You

You, gentle reader, are undoubtedly a trustworthy and trusting person, and I applaud those virtuous qualities. But I've seen many kind souls get scammed, hoodwinked, robbed, and otherwise mistreated precisely because they didn't cultivate an appropriate level of cynicism.

Take, for example, 12-year-old me. After years of pleading and nagging my parents, I finally got that shiny new 10-speed bike I'd always wanted. It was my favorite thing in the world. Then someone opened our unlocked garage in our safe, friendly, suburban neighborhood and stole that bike. It had never occurred to me that such a thing could happen. I was crushed by the loss, but I also learned the importance of using good locks—all the time, even in "safe" surroundings.

When it comes to your passwords, you mustn't ignore the fact that a great many people and machines are constantly probing the world's computers and networks for weak passwords. They probably aren't after *you* personally, but because they don't discriminate, it's in everyone's best interest to take appropriate actions.

Just as you always (I hope) lock your home or your car when you're not in it—just in case—you should always take measures to prevent others from learning or guessing your passwords. Like bikes stolen from unlocked garages in safe neighborhoods, passwords are hacked all the time because their owners didn't take reasonable precautions.

But not all precautions are equally wise. Be skeptical when someone says, "I have this easy but foolproof way to create and remember passwords without any software. Just ____." As I discuss shortly, in Timeworn Tricks, hackers know all these little tricks, too.

Threat #2: Guessing

I already mentioned that someone who knows you (or can learn your personal details, perhaps with a few web searches) has a leg up in guessing your passwords. But even a hacker probing random accounts for password vulnerabilities can often guess your password. That makes guessing a significant threat to just about everyone.

Every few weeks or so, a major security breach occurs somewhere, and millions or even billions of passwords are made public. These lists reveal a great deal about people's password habits, and the results are both interesting and disturbingly consistent. Time and time again, the same passwords are the most frequently used. Although the positions vary by list, the top 100 virtually always include such favorites as `password`, `123456`, `football`, `iloveyou`, `dragon`, `letmein`, and `qwerty`. A sample list of common passwords, updated every year, is Top 200 Most Common Passwords from NordPass.

Needless to say, hackers who want to break into people's accounts without any personal knowledge of their targets try all these passwords first. And I don't mean only the top 200 passwords. They'll run through millions of the most common passwords—in seconds. A scary percentage of accounts are vulnerable to such attacks.

> **Note:** Some systems introduce delays between password attempts or lock out users after a certain number of incorrect attempts, but you shouldn't let these safeguards give you a false sense of security. I explain why just ahead, in Threat #3: Brute-Force Attacks.

Hackers have lists of not only common passwords but also words in the dictionaries of virtually every language, names, and other strings from history, literature, and popular culture (`WarOf1812`, `Psalm23`, `NCC1701`, and so on). All these terms—millions of them—can be tested rapidly against most password systems in an automated procedure known as a *dictionary attack*. Modern dictionary attacks don't stop there; if they're unsuccessful after running through the passwords in their list, they try common variants such as reversed order and unusual capitalization. These sophisticated *cracking* (that is, guessing) algorithms can uncover most passwords with shocking speed. And hackers are constantly improving their tools and techniques.

Threat #3: Brute-Force Attacks

If simpler methods fail—and a password is deemed valuable enough—a hacker may resort to a *brute-force attack*, in which a computer pro-

gram systematically tries every possible string of characters as a password until it finds a match.

Note: Technically, the program may be computing *hashes* for potential passwords and comparing them to the target account's password hash—see the sidebar About Hashes and Salts—but it works out to roughly the same end result.

Brute-force searches are guaranteed to succeed, given enough time. If the password is long enough and complex enough (see the next chapter, Learn About Password Security), that time might be centuries or millennia—long enough that your password is safe for all practical purposes. But progress marches on, as do the technologies available for brute-force attacks, so a password deemed "safe for all practical purposes" a few years ago might seem laughably insecure today.

I've made the mistake of overestimating password security against brute-force attacks myself. In an earlier book on passwords (originally published in 2006 and last updated in 2010), I wrote:

> *If the thief used a very fast desktop computer that could check ten million passwords per second, and if your eight-character password contained alphanumeric and punctuation characters…it could take up to 21 years for the computer to guess it…. If the thief had a large supercomputer (or a thousand fast desktop computers networked together), this time would drop to a little more than a week. But if you added just one more character to the password, even a supercomputer would need nearly 4,000 years to figure it out! … So for all practical purposes, a nine-character password with alphanumeric and punctuation characters is effectively uncrackable—but only if it's random, because thieves are likely to try more predictable passwords before deploying a brute-force attack.*

Today, my earlier claim seems downright quaint. As far back as 2012, 8-character passwords containing upper- and lowercase letters, digits, and symbols could in some cases be cracked by brute force in *five and a half hours*—a *wee* bit faster than my previous estimate of 21 years.

With modern hardware and algorithms, the time required for a brute-force attack is even smaller. (This table, updated annually, gives you a quick way to check on the brute-force cracking time for passwords of various lengths and character type combinations—assuming, in this case, an encryption algorithm called bcrypt.)

Note: The time required to crack a password of given complexity depends on the specific encryption algorithm used. As websites and password managers increasingly shift to stronger, more modern algorithms, cracking time increases, but since you generally can't know which algorithm is used, it's safest to assume the worst case.

How did password cracking get so much faster? One reason is a common technique that uses GPUs (graphics processing units), which are much faster than CPUs at these types of computations. Even an off-the-shelf GPU in a run-of-the-mill PC combined with free software can easily check billions of passwords per second (read How A Cheap Graphics Card Could Crack Your Password In Under A Second from back in 2011—and imagine how much more powerful such hardware is today). Add more or better cards, and the rate goes up dramatically. Another reason is that modern cloud computing services let anyone assemble a virtual army of computers to unleash on any task they can imagine, without actually having to buy, house, or set up any physical equipment. (Malware purveyors can also employ an illicit, covert network of unsuspecting computers—a *botnet*—to do the same thing.)

Now, I know what you're probably thinking (because I've heard this counterargument many times): "You claim that a computer can check a zillion passwords per second, but whenever I log in to a computer or website, it takes several seconds even if I get the password right. If I get it wrong, the delay increases, and if I get it wrong more than a few times, it locks me out. Don't those safeguards make the system immune to automated attacks that require many rapid guesses?"

Sorry, not so much.

For one thing, automated attacks often use a technique called credential stuffing to work around these safeguards. With readily available

tools, attackers can make login attempts appear to come from a wide range of IP addresses and browsers simultaneously—bypassing safeguards triggered by repeated login failures from a single device.

For another thing, the most successful brute-force attacks don't go through the front door, as it were. Rather than taking place *online* (by which I mean being used against a live system, with safeguards in place), they're done *offline* (bypassing most security measures and operating directly on a data file). This story can play out in several different ways, so let me give you just one example.

Suppose there's a site we'll call YouFace (hat tip to *30 Rock*) that stores login credentials for many users. Your password isn't stored in a plain text file. (YouFace would never use such an unsafe practice, although utility companies or social media giants might.) It's in an *encrypted* database file! (For this illustration, we'll suppose that the passwords aren't individually protected with *hashes* and *salts*—as they often aren't—see the sidebar About Hashes and Salts, next.) Only a handful of system administrators at YouFace have the password to decrypt that file and see the passwords in it. The data is pretty safe.

One day, the encrypted file gets out. Someone hacks into the network, or a company laptop is stolen, or a disgruntled employee leaks the file. It doesn't matter how, but the data gets into a hacker's hands.

The file is still encrypted, of course. But now the person who has that file can play with it offline—avoiding the pesky delays, Captcha checks, lockouts, and other barriers that might appear on the web. A hacker can apply considerable computing power to check many passwords per second in an attempt to crack the single password that protects the file. Once that password has been discovered and the file has been decrypted, *all* the individual passwords inside it are there in the clear.

Offline attacks can take many other forms, and it's not worth getting bogged down in the details. All you need to know is that there are various ways a hacker may gain direct access to encrypted data (even on your own personal computer), making all those online safeguards irrelevant. But if you make every password unique, then even if one is uncovered in an attack like this, at least the damage will be contained.

About Hashes and Salts

Many systems—sadly, not all—store each password not as clear text but as a *hash*. A hash is the result of a one-way mathematical function that turns one string into another. For example, if you apply a certain hash function to the password `baseball`, you get `e21e15b460f26696ffbf4a41c0ddde4f`. That string, and not the actual password, is what the system stores.

Later, when I enter my password `baseball`, the system runs it through the same hash function and checks the result against what it previously stored. If it matches, it knows I've entered my password correctly. But—and this is the crucial part—because a hash function is one-way, the system can't turn `e21e15b460f26696ffbf4a41c0ddde4f` back into `baseball`. That means no one who sees the hash knows what your actual password is, only whether the password you entered today matches what you entered when you signed up.

Now, say that list of hashes falls into a hacker's hands. That hacker can't use the hashes to log in, doesn't know the passwords, and can't convert the hashes back into passwords. But the hacker can generate their own table (called a *rainbow table*) of hashes and their corresponding plain-text passwords by adding a hash function to a dictionary or brute-force attack. And, wouldn't you know it, one of the very first hashes they generate this way is `e21e15b460f26696ffbf4a41c0dddde4f`! The hacker knows that they used `baseball` to create that hash, so they now know that's my password. In fact, enormously long rainbow tables are freely available online, giving hackers easy access to precomputed hashes and the passwords used to generate them.

There are three main ways to address this vulnerability. On the provider's end, the system designer can choose a stronger hashing algorithm (some are much better than others). The designer can also add a *salt*—a small chunk of random data—to each password before hashing it. Salted hashes are much more secure, because an attacker must try all possible salt values with each password when generating a hash list, a process that's often impractically time-consuming.

On the user's end, the best way to reduce one's vulnerability to rainbow tables is to choose a long, random password, because the probability that such a password will have a precomputed hash is quite small. A strong enough password can usually survive an offline attack against a hashed list, even if the hashes are not salted.

Password Cracking for Everyone

If you have the idea that password cracking is an elite sport undertaken only by trained professionals with years of experience, you might find Nate Anderson's 2013 article How I became a password cracker, at Ars Technica, enlightening. Nate shows how anyone can learn to crack passwords, with an ordinary Mac or PC, in a matter of hours. This article is also interesting in that it provides details about how real-world cracking algorithms work. It nicely illustrates several of the threats I discuss in this chapter, and the major insights from the article are still applicable today, even though some of the technology is different now.

Threat #4: Theft/Hacking/Sniffing

The next group of threats involves other people being able to see and steal your password without your knowledge. Even if you have the world's longest and most complex password, it's worthless if someone walks up to your desk while you're gone and reads it from a sticky note attached to your monitor. I'll assume that you don't have any of your passwords hanging up where other people can see them. (If you do, the paper shredder is right over there. I'll wait until you get back.)

Someone can steal your password in all sorts of ways, such as:

- **Physical theft:** Someone ransacks your office or steals your wallet to find a piece of paper on which you've written a password.
- **Hacking:** Someone hacks into your computer (perhaps by way of malware you accidentally downloaded), finds your password there, and copies it. Or, worse, someone hacks one a site where you happen to have an account and steals your password—plus those for every other user.
- **Keystroke logging:** If you use a public computer, there's a chance someone might have installed a *keystroke logger*—a piece of hardware or software that records every key pressed. If your computer has been infected with malware, that could also log keystrokes. Later, when the bad guys examine keystroke logs, it's easy to pick out passwords. One way to guard against keystroke loggers is to use

a password manager (see Use a Password Manager for Everything Else), which eliminates the typing step altogether.

- **Sniffing:** Someone monitors network traffic (typically public Wi-Fi networks with no password or insecure WEP passwords) and looks for username/password combinations as they're sent from your computer to a server; this is called *sniffing*.
- **Looking over your shoulder:** If you're not careful when you log in to your laptop or a secure site in a public place (a coffee shop or an airplane, say), someone nearby can watch your fingers as you type your password. (I'm not kidding! This happens all the time.) You can reduce this risk by using a password manager (see Use a Password Manager for Everything Else). But beware: this problem becomes exponentially more dangerous when you're talking about a smartphone with a shorter (and usually numeric) passcode. I explain why a bit later; see What About Mobile Device Passcodes?.

And these are just a few of many possibilities. Most of these vulnerabilities are directly related to your proximity to other people. If you work in an open office with lots of other employees, use your laptop at a library or on a plane, or live in a densely populated urban area with lots of Wi-Fi networks, you're far more susceptible to some sort of password theft than if you use your computer only in a remote or physically secure environment. However, regardless of your location, it's worth thinking through ways in which a stranger might be able to see a password, and take appropriate precautions (as discussed throughout this book).

Threat #5: Social Engineering

I've told you about some of the technological means someone can use to discover your password. But often, there's a far easier approach: getting you (or someone else) to *tell* them! When someone verbally manipulates you into giving up personal information such as a password (directly or indirectly), that's called *social engineering*.

Spy stories are full of this sort of thing. You've probably seen it on TV a hundred times: the secret agent befriends or seduces the target and

then, with the most innocent-sounding motives, gradually wheedles out some top-secret information.

But spies aren't the only people who excel at social engineering. So do 16-year-old hackers. (For a chilling example from 2012, see Mat Honan's article in Wired about his experiences, Kill the Password: A String of Characters Won't Protect You.) Every time you receive a "phishing" email—you know, the ones pretending to be from your bank or PayPal or Amazon asking you to verify your information so you'll go to a fake site and enter your real password (see the sidebar Avoid Password Phishing Scams, later)—you're being subjected to a form of social engineering. It might also come as a phone call, a casual inquiry while waiting in line at Starbucks, or in many other ways.

Don't misunderstand: I'm not saying a hot CIA agent is going to make out with you and then say, "By the way, what's the password to your bank account?" Social engineering is usually much more subtle and indirect. Someone might instead try to find out your first pet's name, or the name of the street where you grew up, because those sorts of things are often used as security questions for resetting passwords (read Understanding Security Questions and Reset Procedures). And they may not even try to get that information directly from you, but from a friend, employer, data broker, or other source. (In extreme cases, someone might try using a wrench to make you reveal your password—admittedly, not exactly *social* engineering.)

Tip: I say quite a bit more about dealing with threats such as these—for example, reducing the amount of your personal data available to data brokers—in my book *Take Control of Your Online Privacy*.

Realistically, most of us don't have such valuable password-protected assets that we need to worry about being *personally* targeted by social engineering. This is more of a worry for celebrities, politicians, and other people with considerable power, money, or influence. And what if you are such a person? Social engineering is extremely hard to guard against, because you're unlikely to recognize it when it happens.

The best way to prevent social engineering from revealing your passwords and the answers to your security questions is *not to know them* (because they were randomly generated and securely stored). I cover all this later, in Apply Joe's Password Strategy.

Timeworn Tricks

Over the years, I've encountered many clever techniques for manually creating and remembering seemingly strong passwords. I've even promoted a few myself. All sorts of little tricks are supposed to give you that fantastic combination of an unguessable, random-looking password that, thanks to a mnemonic clue, you'll never forget.

Well, forget about them.

I'm sorry to break this to you, but people who crack passwords for a living—I'm including professional cryptographers and hackers—are smarter than you, and smarter than me. Not only do they have incredible computing power at their fingertips and serious mathematical skills, they also know how people think. That article you read in *Wired* or in a discussion forum about how to create great passwords? They read it too, and the first thing they did after reading it was to enhance their algorithms to accommodate it.

Let me give an example I've read about in a few places on the web—a technique geared mainly toward Mac users, because of how Mac keyboards handle special characters: take any ordinary word, such as `football`, and type it while holding the Option key. You'll get, in this case, `ƒøø†∫å¬¬`. How's that for random? It's not in any dictionary, should be completely impervious to guessing, and yet is easy to type. Right? Nice try, but it would take any experienced programmer about 30 seconds to add the code to check Option-key versions of everything. Yes, that extra check would take a lot longer, so this technique might slow down an attack, but it's not a panacea. (And, if you ever have to enter your password on a PC or iPhone keyboard, *good luck*!)

Here are a few of the other common tricks I urge you to forget:

- **Simple transformations and substitutions:** Spelling a word backward? Swapping the first and last letter? Toggling the case of every other character—or only the vowels or consonants? Adding a number at the beginning or end? These and dozens of other ways of modifying passwords are already built into freely available cracking software. The same goes for "leet" (or "1337")—a technique in which various letters are substituted with similar-looking numbers or characters. For example, you may find the word `password` rendered as `p455\/\/0rd`, but don't be misled into thinking that's any stronger.

- **Keyboard patterns:** Yes, `qwerty` is a very common password, but so are other patterns of keys on a keyboard, such as `lkjhgfds` and `tgbyhnujm`—even though they *look* random. Cracking algorithms can check hundreds of these patterns in the blink of an eye.

- **Reusing passwords:** I've mentioned this, but I told you I'd be repeating it, and I'm a man of my word. Some people think reusing a password is fine as long as that one password is incredibly strong, but that logic breaks down as soon as someone steals or hacks that one password! So, for maximum protection, don't use the same password for multiple sites or services.

- **Padding, a.k.a. the "Haystack" method:** A method popularized by Steve Gibson involves constructing passwords out of short, easy words (slightly obfuscated with uppercase characters, numbers, and/or symbols) and padded out to an arbitrary length (such as 24 characters) with a pattern of your choice—say, a string of commas or `*-&` over and over again. The logic is that you get passwords that are easy to remember but are still hard to guess because they have high entropy (see All About Entropy).

  With all due respect to Steve, I think this is a bad idea.

  First, it's one thing to use this method for a single password, but if you have to remember dozens or hundreds of unique passwords like this, it'll be impossible unless you "cheat" by including in the password a hint about the service it goes with. That decreases its securi-

ty considerably, because anyone who discovered one of your passwords could reverse-engineer your system.

Second, and more important, is the fact that any cracking algorithm worth its salt (so to speak) would test all such patterns long before moving on to purely random passwords—and if the hacker has any clues about how you construct passwords, it'll go that much quicker. (Remember, hackers read that webpage too!) So, I think it provides an inflated sense of security. Besides, there are much easier ways to create and remember passwords, as I discuss later.

If you've been using any of these or other common tricks, don't feel discouraged. You should make some changes to improve your security, but they won't be too hard. If you follow my suggestions in Apply Joe's Password Strategy, you'll have only a handful of passwords that must be both complex and memorable. When we get to that point, I'll offer several suggestions that should put you on the right path.

Usernames and Passwords: an Outdated Model

The biggest issue with passwords is that they offer an outdated solution to an increasingly complex and high-stakes problem. Securing a computer resource with a username and password made reasonable sense decades ago, when most people had very few accounts and when security threats were both fewer and less serious.

But today, that security model seems silly. A few of the reasons:

- Having a few passwords is one thing; having hundreds or thousands is another. Yet every time I turn around, another site asks me to create a new password. This situation puts users in an untenable position: Either follow the manageable but insecure route of reusing simple passwords, or put time, effort, and money into creating and using a secure but inconvenient system.
- No universal standard exists for password security. So, while one company may exercise enormous care with your data—enforcing

strong passwords and using salted hashes, strong encryption, and rigorous internal security policies—the next site you visit may limit you to an 8-character password that's stored in a plain text file on somebody's laptop. *But you have no way to know*.

- Passwords, even strong ones, don't identify a person uniquely. A password is like cash—whoever holds it can use it, so if it's stolen or found, it works as well as if it's in the hands of a legitimate owner. Multi-Factor Authentication helps somewhat with this problem, but not infallibly—and at the cost of greater inconvenience.
- Given the number of people on the internet, economies of scale make password cracking (and theft) highly profitable—even when it requires enormous computing resources to pull it off.

If using usernames and passwords is an obsolete system, what's the solution? Is there a better way to accomplish the same goal? After decades of reading articles about various plans to "kill passwords" that turned out to be little more than wishful thinking, I'm happy to say that a modern replacement for the tired old username-and-password system finally exists. It's called a *passkey* and you can start using it today! I tell you all about it in the chapter Authenticate Without Passwords, later.

Passkeys are great because they're vastly more secure than passwords *and* far easier to use! The problem is, as of early 2025, only a handful of sites and services have begun offering them as an alternative, and even then, they work only with recent operating systems and browsers. Plus, even sites that do offer passkeys almost invariably require a backup authentication method, and that would be…you guessed it: usernames and passwords.

So, we'll get there, but it will take a while. In the meantime, most alternative authentication schemes you're likely to encounter involve either or both of two fundamental concepts: biometrics and authenticator devices. I want to acquaint you with both of those, because not only can they improve the security of your existing passwords, they also factor into your future use of passkeys.

Biometrics

Biometrics refers to methods of authentication that rely on measurable details about your body such as fingerprint, retina, or iris scans; facial recognition; hand geometry; or even the pattern of your heartbeats. The security that comes from the uniqueness of each person's body—plus the fact that biometric hardware is becoming smaller, cheaper, and faster—has made biometrics increasingly prevalent and useful. For example, many smartphones, tablets, and laptops have built-in fingerprint scanners that can be used in lieu of passcodes for unlocking the device itself and various apps and services that run on the device. Other devices, such as newer Apple iPhone and iPad models, use a sophisticated 3D facial recognition system called Face ID for the same purpose. And Windows Hello on a compatible Windows 10 or Windows 11 computer or tablet enables you to authenticate using either face recognition or a fingerprint scan.

But in most cases, biometrics is an alternative or supplement to passwords, not a replacement. For example, if you burn your thumb so your fingerprint can't be read, you can still enter a password to unlock your MacBook Pro, and if you're wearing a mask, you can still use a passcode (or an Apple Watch you've already unlocked with a passcode) to unlock your iPhone. Some Multi-Factor Authentication systems can use a biometric factor *in addition to* your password to prove your identity with extra certainty. But using biometrics alone—without any fallback or secondary option—is almost unheard of. That's a good thing; it doesn't take much imagination to picture gruesome tactics that might enable an attacker to overcome security based solely on a body part.

> **Note:** Later, when I talk about passkeys in Authenticate Without Passwords, I describe a scenario in which biometrics *appears* to log you in all by itself, but that's an oversimplification. When it comes to passkeys, it's really a trusted device that's doing most of the heavy lifting, and biometrics merely confirms that you're the device's owner.

Furthermore, there will always be some devices that require authentication but that can't or shouldn't use biometrics. (Do you really want

your TV to scan your eyes so you can log in to Netflix? There's no reason it couldn't be done, but it sounds pretty creepy to me!)

So although biometrics is increasingly useful as both a security measure and a convenience feature, by itself, it isn't a plausible replacement for passwords.

Authenticator Devices

If biometrics isn't the magic bullet to replace passwords, what about an object you can carry in your pocket or on your wrist or keyring? If your computer or phone can sense the presence of this unique device nearby (using Bluetooth, for example), it should be able to log you in without requiring you to enter any password at all. That's the theory behind a growing number of devices (and mobile apps that run on devices you already have). I refer to these technologies collectively as *authenticator devices*.

I've seen quite a few different spins on authenticator devices. Each one promises greater security and convenience than the last, and some are pretty cool. Like biometrics, authenticator devices don't normally replace the underlying password mechanism as such; rather, they provide an automated way to log in without having to type, paste, or otherwise enter passwords you've already set up. If your authenticator device were to go missing or stop working, you could still log in by typing your password.

That major qualification aside, let me offer some examples of authenticator devices that can reduce the need to enter passwords manually:

- **Apple Watch and iPhone apps:** Without any additional software, your Apple Watch can unlock an appropriately configured Mac—just touch a key on the Mac and, as long as your watch is unlocked and on your wrist (as confirmed by heartbeat detection), your Mac unlocks automatically. If your Mac is running macOS 10.15 Catalina or later, your Apple Watch can also authenticate system requests and certain third-party apps. For those without an Apple Watch or with older Macs that don't support this feature, an iPhone app called Knock lets you knock twice on your iPhone to

unlock a nearby Mac running a special companion app; MacID does something similar but uses Touch ID—and includes the option to log in on your Mac by tapping a pattern on your trackpad or Magic Mouse. (Both Knock and MacID can also be used with an Apple Watch.)

Note: You must configure your Mac to use an Apple Watch to unlock; it's not set up this way by default.

- **Everykey:** Everykey uses a tiny Bluetooth fob in conjunction with software running on your Mac, PC, or mobile device. Bring the Everykey close enough, and your device unlocks; move away, and it locks again. But Everykey is even more ambitious than that. Not only can it log you in to your device, it can also fill passwords in your web browser and other apps. (The passwords themselves aren't stored on the device; the companion software generates and stores them, and the Everykey itself stores only the encryption key that unlocks the software.) It's even designed in such a way that—given hypothetical future support from third-party developers—it could open the door to your house or start your car. If your Everykey is lost or stolen, you can remotely disable it.

  I haven't yet seen an Everykey in person and can't comment on how well it works. But I do note that it has some significant limitations. For example, it can unlock iPhones only if they're jailbroken—a major security risk that I don't recommend to anyone. (The mobile Everykey app essentially functions as a password manager on devices it can't unlock, which is still useful, but not in the same way.) And the developer has been promising support for unlocking doors and cars for years, but I'm increasingly skeptical about that.

One question you should consider when looking at any authenticator device is how easy it would be for someone *else* to use it (ideally, it should be as difficult as possible). Unlocking a Mac with an Apple Watch requires that you first unlock the watch with its passcode (and that it remains on your wrist thereafter), making it reasonably secure. On the other hand, Knock and Everykey, which require no additional authentication after initial setup, assume that their respective proximi-

ty sensors (your iPhone or key fob) will remain safely on your person at all times. That's their biggest weakness: anyone who got hold of the object could unlock your computer. (In the case of the Everykey, at least you can disable it remotely.) In addition, their range is highly variable, meaning your Mac could be unlocked even if you're not close enough to see or control it. So, think carefully before using any of these products, because their added convenience can increase your vulnerability.

Devices that require biometric readings (for example, Apple Watch, and MacID with Touch ID enabled) are far less likely to be useful to an attacker if they're out of your immediate possession.

What About Magic Links?

Another type of passwordless login, which has gained a bit of popularity in recent years, is the *magic link*. The basic idea is: when it comes time to log in, you supply your username but *not* a password. Instead, the site or service sends you a single-use link via email or SMS, and you click or tap that link before it expires (usually within a few minutes or so) to log in. Slack, Tumblr, and Medium are among the services that use this mechanism.

Magic links solve the problem of having to remember and enter passwords. However, depending on the circumstances, they may or may not be more convenient than passwords, and they're significantly less secure.

The first problem is that any number of factors could delay an email or SMS message, or prevent you from seeing it. (Magic link messages are often marked as spam, for example.) Waiting to receive a message before you can log in is not a good user experience.

More serious is the security problem: a magic link is only as secure as your email account or SMS service—and neither of those is inherently secure at all. If someone got access to your email account, stole your phone, or intercepted an SMS message—all of which are frighteningly easy for a sophisticated attacker—they could easily log in to your accounts using magic links and do all sorts of damage.

In short, magic links are a poor, if not downright misguided, solution to the password problem. Passkeys, by contrast (see Authenticate Without Passwords), have none of those shortcomings, making them a much better and safer solution.

A Future Without Passwords?

By themselves, techniques such as biometrics and authenticator devices merely address the inconvenience of entering passwords by asking you to do something else (which may or may not be more convenient or more secure). But there's a difference between a more convenient password entry method and a true replacement for the whole, awful username/password model.

The real password killer is the passkey, and I tell you all about it in Authenticate Without Passwords. I've begun using passkeys on my

own devices here and there, and where they're supported, they're great! But it will be up to each site and service that currently uses passwords to implement the new system. That could take anywhere from many years to forever.

While I can't make the underlying problem go away completely or soon, I do hope that by the end of this book, you'll feel that the symptoms are under control. And you won't need a bleeding-edge gadget to achieve that control, either.

Learn About Password Security

We begin with a brief lesson on password security. I want to keep it short, so I won't go into tremendous detail about encryption algorithms and cryptographic mathematics, and I'm going to do a bit of hand-waving when we get to the geekier concepts (and refer you to Appendix C: Calculate Password Strength if you're genuinely interested in the details). But I think it's important to have a basic grasp of the principles of password usage so you know what you're up against, and why simple-sounding solutions are often extremely unwise.

And, even if you were well-versed in password security basics a few years ago, you should be sure to read about Multi-Factor Authentication, which has become increasingly important.

What Makes a Good Password?

To put it simply, a good password is one that you won't forget but that no one else (human or computer) can guess. Behind that straightforward description are two knotty, interconnected problems:

- **Guessability:** Most people have an unrealistic idea of what "guessable" means. You might imagine that no one could connect the password `ninjaboy` with you, but the computer I'm using right now could figure that out before I finish typing this sentence. As I explained in The Major Threats, even if a human who knew everything about you would never guess your passwords, sophisticated cracking algorithms may be able to figure them out unless you take steps to thwart them (discussed at length just ahead). To avoid that risk, your passwords should be far more complex than you might think.
- **Memorability:** If you can't remember a password, it's useless. As a password's complexity (and thus its strength) increases, its memo-

rability tends to decrease. Let's face it, `iYb48nzJ#;sEoR` may be vastly stronger than `ninjaboy`, but it doesn't exactly trip off the fingertips.

Creating memorable but unguessable passwords—and not just one or two, but potentially hundreds—may sound like an intractable problem. But hang tight; we'll get to a strategy shortly.

All About Entropy

Let's quantify this vague notion of guessability. In ordinary speech, the word *entropy* means disorder, randomness, or unpredictability. Cryptographers use the term *entropy* to refer to a mathematical approximation of a password's complexity based on the method used to create it. A password with higher entropy is harder for a person (and, more importantly, a machine) to guess. So, for passwords, higher entropy is a very good thing.

Note: Cryptographers measure password entropy in *bits*; a larger number of bits means higher entropy. If you're interested in learning how entropy is calculated—and why it's possible to get numerous conflicting entropy values for a single password—consult Appendix C: Calculate Password Strength. The entropy values I mention in this chapter are derived from the zxcvbn password strength calculator.

But how does higher entropy (or complexity) help make passwords harder to guess?

You already know that cracking algorithms can check billions of passwords per second in an attempt to figure out what yours is. But even brute-force searches don't go in alphanumeric order. (If they did, then `zzzzzzzzzz` would be much stronger than `aaaaaaaaaa`, but it isn't.) Instead, cracking software identifies common patterns of characters that few humans would notice. It uses this information to test more likely passwords before less likely ones, reducing the average time it takes to produce a match. Because higher-entropy passwords are less likely to be used than lower-entropy passwords, a brute-force search tends to take longer to find them.

In other words, it's like the difference between finding a particular fish in an aquarium versus finding a particular fish in the ocean—it could happen, but the odds against it are immense. High-entropy passwords thwart cracking algorithms by giving them an unreasonably large search space to explore—they might churn away for decades or even centuries, on average, before they find a match. Unless they get very lucky indeed, they'll give up long before they find your password.

Note: The art of cracking passwords involves a lot more than throwing a single algorithm at an encrypted string. You can learn more about common cracking techniques in the fascinating and disturbing 2013 article Anatomy of a hack: How crackers ransack passwords like "qeadzcwrsfxv1331".

Now that I've explained why you want high-entropy passwords, how do you get them? You may be surprised to hear that when it comes to password entropy, "complex" doesn't necessarily imply "complicated." There's more than one path to high entropy. To oversimplify slightly, the major factors that influence a password's entropy are its length, the size of its character set, and its randomness. Let me explain:

- **Length:** Say you have a 4-character password in which all the characters are lowercase English letters. You know that there are 26 possible values for each character, so the total number of choices is 26 × 26 × 26 × 26, or 456,976. Add another letter and you get to multiply that total by 26 again, for 11,881,376 possible choices—and so on. So making a password longer by even one character increases its entropy exponentially, because it will take that many more attempts, on average, to guess it. Increasing its length is the easiest and most effective way of increasing a password's entropy.
- **Character set:** Continuing with the same example, what if your password can contain both uppercase and lowercase letters? Now there are 52 possible values for each slot, so a 4-character password has 7,311,616 variations (52 × 52 × 52 × 52), and a 5-character password has 380,204,032. That's a huge increase over the number you get when each character can have only 26 possible values. Throw in the digits from 0–9 and you have 62 possibilities per

character. Add a dozen punctuation characters and you have 74, thus raising the number of possibilities for a 5-character password to 2,219,006,624. (Don't forget, though: a computer can check all two billion of those passwords in less than a second!)

The moral of the story is that the larger the number of possible characters a computer has to try for each slot in a password, the longer it will take to guess it. That's why so many sites require you to use upper- and lowercase letters, digits, and punctuation—to force the target character set to be as large as possible.

- **Randomness:** The string `horse` is, as we've seen, one of 11,881,376 possible 5-letter lowercase passwords. And yet its entropy is far lower (about 9 bits) than another 5-letter password—say, `dcxuw` (about 17 bits). That's because randomness is another factor influencing entropy, and `dcxuw` is random, whereas `horse` is not.

  For the purpose of this discussion, think of randomness as an "absence of a discernible pattern." Dictionary words follow patterns. So do pronounceable strings that aren't words (such as `glavondi`), although the patterns are more subtle. But as I pointed out earlier, password-cracking software can recognize all sorts of other patterns that a casual human observer might never spot, including the most common methods of constructing seemingly random passwords.

  Creating truly random strings is nearly impossible for a human, and it's difficult even for a computer. Most of the strings your computer produces are *pseudorandom,* which is to say largely but not perfectly unpredictable. But they're invariably more random—and therefore better contributors to a password's entropy—than anything you'd think up yourself.

Note: Research by Microsoft found that hackers rarely even attempt brute-force attacks against passwords longer than 10 characters or containing special characters, regardless of their actual entropy. That's encouraging news, but don't let it lull you into complacency, because the story could change at any time.

Now, let's put that all together. Let's say we're shooting for some arbitrary level of entropy. I'll pick an extremely low value of 20 bits as a target for the sake of illustration. (In reality, I recommend using passwords with a much higher entropy—say, 75 bits or more.) How might we get that?

Well, `RbJwEkb`, a random 7-character string with both lower- and uppercase letters, has 23.25 bits of entropy (according to one calculation, at least; see the note just ahead). But we can do even better with a 9-letter, all-lowercase dictionary word, `mesomorph` (27.26 bits)—what it loses in randomness and character set size, it makes up for in length.

> **Note:** For simplicity, in this chapter I'm using an entropy calculation based on a relatively small dictionary. One could argue that *any* dictionary entry, no matter how long, should be considered to have no more than 16–20 bits of entropy, and I explain why in Appendix C: Calculate Password Strength. But the point remains: longer can be just as good as (or better than) random-but-short.

That's essentially the point of the wonderful and famous Password Strength comic from XKCD—a shorter but more complex and therefore less memorable password (`Tr0ub4dor&3`) has far less entropy than a longer but more memorable password (`correct horse battery staple`).

Now, before you get all excited about ditching random gibberish passwords for simple, all-lowercase phrases, I should point out a few downsides of that method:

- Many websites and other services that use passwords disallow space characters. So you might have to run your words together (as in `correcthorsebatterystaple`) or separate them with another character (`correct-horse-battery-staple`).
- To reach a level of entropy I'd consider safe, you'll need a longer string—I recommend at least 32 characters in all (`correct horse battery staple` is only 28) *and* a total of five or more words. Even then, you might be at risk if you choose your own words, which significantly lowers the overall entropy—make sure to use a random method of generation that relies on a long list of potential words.

(For example, there's an online password generator based on the XKCD method; the strong-but-labor-intensive Diceware method; and 1Password's Memorable option, which is conceptually similar to Diceware but more secure.) But…

- Just as most sites disallow spaces, many have limits on password length—you're often restricted to passwords shorter than 28 characters, let alone 32 or longer. More likely than not, they'll also require at least one uppercase letter, one digit, and one punctuation character. And, some even forbid including *any* dictionary words in your password!

> **Note:** The password for an Apple Account (formerly called Apple ID and used for iCloud, Apple Music, the App Store, and so on) can have up to 32 characters. In some cases, Apple lets you type in more than 32 characters when setting or changing your password, but only the first 32 are ever stored or recognized.

- Yes, it's easier to remember a few words than a string of random characters. But it's one thing to do this for a single password, or even a few, and another thing to do it for dozens or hundreds of passwords. So, at best, it's a technique you might use for a handful of passwords (see Figure Out Which Passwords You Must Memorize).

Despite all those problems, the essential insight is well worth bearing in mind: there's more than one way to skin a password.

> ***Warning!*** Due to a flaw in the way Windows XP handled passwords, even a random 14-character password was completely insecure, because it was stored as two 7-character passwords, each one very easy to crack! This problem was fixed in Windows Vista and later, but if you're still using Windows XP (seriously?), your password must be at least 15 characters long to avoid this flaw.

Should Usernames Be Unique and Random Too?

Several people have asked me whether one should take the same care when choosing usernames as when choosing passwords—in other words, not using your regular email address or a username someone might guess is yours. After all, so the argument goes, most sites require both pieces of data, and if an attacker doesn't know your username, that's another hurdle to overcome in order to access your account. Using an email address everyone knows is, in a sense, handing out one of a pair of keys to your digital door.

I don't disagree with that logic; putting more barriers between you and the bad guys can't hurt. But it also makes more work for you. If you use a password manager, as I discuss later in this book, it's not much extra effort to have it generate fake usernames, just as it would for passwords—although I know of no password managers that can generate a random username and password in a single step. (And, if that username is something other people will see—I'm thinking of discussion boards and the like—you may prefer to represent yourself as your actual identity, not as a pseudonym.)

Some sites, however, require that your username be an email address. If you're an Apple user, the Sign in with Apple feature and, for paid iCloud+ subscribers, Hide My Email, let you create pseudonymous credentials on the fly, with randomized email addresses that forward messages to you while hiding your real identity. These and similar services also serve the function of making your username that much harder to guess. But if you aren't using such a feature and your username must be an email address, you'll have to not only create a new, non-public email account but also check it regularly, since that's where verification and reset messages will go.

Finally, keep in mind that although passwords are routinely encrypted on web servers, usernames often aren't. So no matter what you do, you can't guarantee that a hard-to-guess username will actually increase your security.

For all these reasons, I personally stick with conventional usernames nearly all the time, but if you have exceptional privacy needs and don't mind the extra inconvenience, you should feel free to take the extra step of making your usernames unique.

Why a Great Password Isn't Enough

No matter what technique you use to create a password, and no matter how high its entropy, other factors can nullify your password's effectiveness. I'll mention a few:

- **The sticky note problem:** It's become a cliché: someone writes a password on a sticky note and puts it on their monitor so they won't forget it…but then everyone else in the office can see it too. You might not do that, but if an unauthorized person can see your password—whether on paper, in digital form, or otherwise—it might as well be `football`, because it's completely insecure. I talk about the care and feeding of passwords in Keep Your Passwords Secure.
- **New wine in old wineskins:** I want to call your attention to a particular instance of the last point—you have fantastic passwords but you store them on your computer in a plain text file or other unencrypted location. If someone steals your computer, hacks your network, or sits down at your desk when you take a coffee break, your great passwords are toast. Keep those valuables locked up; see Use a Password Manager for Everything Else.
- **Screen door on a submarine:** There are cases in which you keep your password totally secure, but the site you've stored it on doesn't. Because of poor security policies or inept programming, your password falls into the wrong hands (see Threat #4: Theft/Hacking/Sniffing). There may also be systemic flaws in the encryption algorithm used to secure your password, or faults in your password manager's password generator that prevent "random" passwords from being as random as they should be. You can't prevent these things, but you can contain any damage that might result by never using the same password in multiple places.

One way to reduce these risks is to use Multi-Factor Authentication, which I discuss shortly.

What About Mobile Device Passcodes?

You use passwords to log in to websites, email accounts, and other online services, and you use them to unlock your desktop or laptop computer. But mobile devices—phones, tablets, smartwatches, and the like—usually prompt you to set up and use a *passcode* instead. Passcodes function just like passwords, but they're typically much shorter and simpler: a 4- or 6-character numeric code is what most people use for most mobile devices.

I can only presume that short, numeric passcodes are the default for mobile devices because longer, more complex passwords are much more difficult and error-prone to enter, especially when using just one hand. From a usability standpoint, that makes perfect sense. However, in terms of security, short, numeric passcodes are *terrible*, because everything I said so far in this chapter about entropy and guessability is still true, even for a phone or watch.

In fact, it's even worse than that. With nothing but your phone and its passcode, you can reset the passwords to most, if not all, of your accounts (including such crucial ones as an Apple Account or Google account). That means a criminal who looks over your shoulder as you enter a passcode and then steals your phone can do the same—locking you out so you can't even reset your access from another device! A Wall Street Journal article in February 2023 described cases where exactly this has happened, causing victims to lose access to their email, photos, and other data—as well as their money—with little to no recourse.

In my opinion, a 4-digit code is too risky for any device. (Even worse: having no passcode at all—which is the case for a shocking number of people.) Most mobile devices have an option to use a longer numeric passcode or an alphanumeric password, and doing so is a good first step. (See this page for iPhones and iPads, and this one for Android devices. Android phones also let you draw a pattern on the screen with your finger in lieu of a passcode.) On an Apple Watch, go to Settings > Passcode and turn off Simple Passcode; then enter your current (4-

digit) passcode, enter and repeat a longer numeric passcode (up to 10 digits), and tap OK.

As you know from reading this chapter, every digit you add to a passcode improves its security exponentially, and using letters makes the passcode even stronger (though harder to enter). But the advantage of a longer, stronger passcode is that it's not merely less guessable, it's harder for someone else to pick up merely by observing you enter it.

Even so, a determined attacker could still—perhaps with the aid of an accomplice or a hidden camera—learn a longer passcode if you enter it in public. So, for your mobile devices, my advice is to enable biometrics (such as Touch ID or Face ID on an iPhone or iPad) so there's nothing at all for someone else to observe when you unlock your device. In situations where you must enter your passcode in public (for example, when restarting a phone), be sure to position yourself in such a way that no one but you can see your device's screen.

Understanding Security Questions and Reset Procedures

What happens if you forget or lose your password, or if you suspect that someone has stolen it? Many sites and services that ask you to create a username and password also require an independent means of verifying your identity, which may involve email messages, phone numbers that can receive SMS messages, and other procedures.

Increasingly, providers ask you to answer one or more so-called security questions. (I've had to answer as many as five such questions for a site.) Usually you can pick from several prewritten questions, and sometimes you can even write your own question.

Security questions serve different purposes. One of these, which I think is somewhat misguided, is to act as secondary passwords. That is, you must know not just two pieces of information (username and password) but three, adding another barrier to password theft without increasing the number of things you must remember. Another purpose

is to reduce tech support calls; by providing an alternative way to verify your identity, providers can make it possible to retrieve or reset a password without human help. And, if a site detects irregular activity and has reason to doubt that a login is legitimate—even though you've entered your password correctly—security questions can help prove that you are who you say you are.

However well-intentioned security questions may be, they fail in several respects:

- Most prewritten security questions have answers that are absurdly easy to find, *assuming you've answered them truthfully*. Anyone with modest web searching skills can probably learn your mother's maiden name, the street where you grew up, and—if you've shared personal details on a blog or a site like Facebook—the name of your pet, your favorite sports team, and other common questions. Security questions may stop some automated attacks, but they're no match for a human. (Sites that let you write your own question are a bit better in this respect.)
- Conversely, if you lie when answering security questions—which I strongly recommend—you now have the burden of remembering that lie and being able to produce it when needed.
- Many security questions are ambiguous, or have answers that change over time. "Where did you go to college?" Which time? And, are you asking for the name of the college or the name of the city? I might not remember which answer I used later. Recently a site asked me a question I'd answered many years ago: "What is the name of your oldest niece?" I typed in the correct answer, but it was rejected. I later figured out the reason—I'd originally supplied the answer before I got married, but my oldest niece today is on my wife's side of the family.
- In some cases, answering one or more security questions—without any further verification—can enable you to reset your password. But if you can do that, so can anyone else. That means the answers to your security questions are even more important than your regular password—yet almost certainly much easier to guess!

- When security questions are used as part of a regular login process, they can prevent your password manager's autofill feature from working (see Sites That Thwart Password Managers, later). Besides the inconvenience, if you can't rely on the extra security provided by your password manager, such as verifying that the domain name hasn't been spoofed (or impersonated by a domain with an almost identical spelling), then you're *less* safe when using such a site.

Tip: McSweeny's has a hilarious, dark take on this topic: Nihilistic Password Security Questions.

Later, in Handle Security Questions, I say more about how to choose and answer security questions. But before moving on, I want to talk about a related issue: verification by email.

Often your email address serves as your username, and when it doesn't, providing a valid email address (verified with an emailed link you must click to confirm that you can receive messages at that address) is usually mandatory. If you forget or lose your password, you can then click a link to have a message sent to the address on file for your account that contains either your password or a link you can click to reset your password and supply a new one.

Having a procedure whereby users can recover or reset passwords is a great idea. But most implementations of recovery-by-email are terribly insecure, because they depend on the security of your email account, which is often poor. If someone got access to your email account, they could plug your address into forms on hundreds of websites, click the "forgot password" links, and use the information that shows up moments later in your phone's inbox to access your accounts. (Sometimes, password-reset links sent by email require you to answer security questions too. That's safer than doing without, but it still doesn't make me fond of security questions!)

You can't control how a provider manages password resets, but you can take steps to make email verification and resets more secure. I discuss these in Manage Email Options.

Multi-Factor Authentication

Because passwords alone have so many security problems, providers increasingly use an additional, entirely different form of identification—another *factor*. Sometimes the use of a second (or third) factor is mandatory, but more often, individual users may opt in (and later opt out, if they prefer). Either way, when logging in requires two or more factors, the process is called *multi-factor authentication* (MFA).

Authentication factors fall into three broad categories:

- **What you know:** A username, password, account number, or other piece of information that you should know—but other people shouldn't
- **What you have:** A smart card, secure token, cell phone, smartwatch, USB key, or other object with unique characteristics that, ideally, is in your possession and under your exclusive control
- **What you are:** Physical characteristics such as your fingerprint, the pattern of your iris or retina, your voice, your gait, your heartbeat, or the geometry of your hand or face—refer back to Biometrics for more details

So, for example, if you need both a password and a fingerprint scan to log in to a certain device or service, that's two-factor authentication (2FA)—a "what you know" and a "what you are." (By the way, 2FA is the only form of MFA the average consumer is likely to encounter—a three-factor requirement is exceedingly rare.) But I want to emphasize that MFA requires at least two of the above *categories*, which is not the same thing as two or more items in the *same* category. Thus, if a login requires both a password and an account number, that's two things you *know*, but still only one *factor*. Similarly, if you need only a cell phone and a secure token to log in, that's two things you *have*—and again, that's just one *factor*.

The whole point of multiple factors is to make it impossible for someone to break into account *only* by stealing one or more objects, or *only* by learning secret information, or *only* by forcing someone to submit

to a biometric measurement. Having to do more than one of these things dramatically increases the difficulty for an attacker, and thus increases the user's security.

Most biometric ("what you are") measurements require specialized hardware—a fingerprint or face scanner, say. Because not everyone has a device like that, people like you and me probably won't encounter biometric requirements for two-factor authentication very often (though we may sometimes see them as *options*). Instead, the second factor you're most likely to encounter is a "what you have" item.

Let's look at a couple of examples, starting with something that sounds suspiciously like "what you know," and, in fact, may sometimes be exactly that—a one-time password.

One-Time Passwords

Suppose a site asks for your username and password—both part of "what you know"—and then it prompts you to enter a *second* password in the form of a numeric code that changes frequently. That code is a *one-time password*, or more specifically a *time-based one-time password* (TOTP). Assuming that code is generated by a unique object you have, it counts as a second factor.

Some banks use *secure tokens* for this purpose. They're keychain-sized devices that display a new six-digit number every 30 seconds. When you want to perform certain kinds of transactions, you must log in normally with your username and password and then enter the number currently shown on the token's screen as a second factor.

A secure token is unequivocally "what you have," but it's inconvenient to carry around a bunch of doohickeys like this on your keychain. So increasingly, providers have turned to other methods to deliver one-time passwords. For example:

- **Authenticator apps:** Most of us have a smartphone, smartwatch, and/or tablet within reach at all times. You can use an app running on one of these devices (or, for that matter, on your desktop or laptop computer) to generate TOTPs. Standalone apps of this sort—

which can do their thing even without a network connection—include Google Authenticator and Authy. In addition, some password managers (notably 1Password and Dashlane) can generate TOTPs for you. (The iPhone/iPad versions of Authy, 1Password, and Dashlane also include Apple Watch support, enabling you to display TOTPs on your wrist.) And, Apple devices can now generate TOTP codes without additional software. In all these cases, you get the same end result as using a secure token, but without the need for an additional device beyond one you were carrying anyway.

A twist on TOTPs that I've seen in some authenticator apps is the use of push notifications and a simple "yes/no" response. For example, you start logging in to a site with your username and password and then—instead of getting a numeric code via SMS and having to type that on your computer—an app on your (already trusted) smartphone receives a push notification that you're trying to log in and lets you tap an Allow button (or something similar). Once you do, the response goes back to the server and the website or service logs you in. Apps that offer this capability include Authy and Duo (for multiple platforms, including Apple Watch). But each site must explicitly build in support for this type of service; it's not as universal as a plain TOTP.

- **SMS:** Another way to obtain TOTPs is to have the provider generate them and send them to you via SMS—again, on the theory that your cell phone (even if it's not a smartphone) is "what you have" and is uniquely under your control.

Note: Some providers let you choose between using an authenticator app and SMS, while others offer only one option or the other.

- **Email:** I've seen a few cases where TOTPs are sent via email rather than SMS. Email is much less secure than SMS for this purpose, because email accounts are easily hacked and they're not tied to trusted devices, but it's still better than nothing.
- **QR codes:** Yet another method is for the site or service to display a QR code (or some other visual element) that contains your TOTP;

you scan this with an app on your smartphone to complete the authentication process.

Unfortunately, when a TOTP is generated and displayed by something other than a standalone device such as a secure token, what you get is not *necessarily* a second factor, even though it appears to be. That may sound counterintuitive, so let me explain by way of a few examples:

- **Authenticator app on device logging in:** Suppose I'm trying to log in to a site on my Mac using 1Password. I unlock 1Password with my master password (something I know), then use a keyboard shortcut or pop-up menu to fill in my credentials for that site. Next the site asks for my TOTP, which 1Password fills in automatically, so all I have to do is click Log In or press Return. During the whole process, I've used only one password in one app on one device to get both a password and a TOTP for a site. It's a single factor.
- **SMS interception:** I'm sorry to say that there are numerous ways in which an SMS message can be intercepted between its point of origin and its destination. If that happens, then the interloper knows your second factor, and only the first factor (your password) is truly in play. So if an attacker already knows your password and then intercepts your SMS code, your security disappears.
- **SMS messages viewed on a computer:** Suppose you use a service such as Google Voice to receive phone calls and SMS messages on your computer. As in the previous bullet point, an attacker with access to your computer would not need to consult a separate device to see your TOTP. Similarly, Mac users with iPhones can optionally enable a feature in the Messages app for macOS that enables it to send and receive SMS messages via your iPhone. If you've enabled this feature, your TOTP codes show up on screen automatically—on the same device on which you're logging in.

The bottom line in these and numerous comparable situations is that the existence of both a "what you know" and "what you have" factor is illusory.

And that is exactly why providers who offer TOTPs via authenticator apps or SMS nearly always refer to their services as "two-step verification" (2SV) rather than "two-factor authentication." You have to go through an additional step, which *does* increase your security, but it's less secure than bona fide 2FA because it's merely two instances of a single factor. Although any security expert will tell you that 2SV is not the same as 2FA (see, for example, The difference between two-factor and two-step authentication by Paul Moore), for the purpose of this book, I will, for the most part, treat the two concepts as equivalent.

Application-Specific Passwords

Let's say you use a service such as iCloud or Google Workspace—either of which requires a username and password to log in—then you enable the service's 2FA (or 2SV) requirement. That's fine for logging in to a website, but what about related apps that let you log in once and then store your password so you can remain authenticated indefinitely? (I'm thinking about things like email clients and calendar apps.) These apps aren't designed to support a second factor or step, and most users would balk at having to log in even once—let alone twice—each time they used such an app.

In cases like these, providers typically compromise by requiring *application-specific passwords*. That is, you'll have to log in to a website (with both factors or steps) and then request a special password, generated by the site, that you can enter in a single app (in place of your ordinary password) to log in using that app. Once you've done that once, the app can remain logged in indefinitely. But you can also go back to the site whenever you need to—for example, if your computer is stolen and you don't want the thief to access your email—and revoke any given application-specific password. (For more details, see the sidebar Use App-Specific Passwords.)

Trusted Devices

When you set up a service to send your one-time passwords via SMS to your phone, you're basically saying: "I trust this device. I trust that it's secure (no one else has access to it) and that it will be available when I need to log in. Therefore, I should not have to authenticate every time I use it." You're designating your cell phone as a *trusted device* for that particular 2FA or 2SV service.

A trusted device could be a smartphone, tablet, smartwatch, car, Bluetooth key fob, or any of numerous other objects. It's something you've set up once (by proving your identity in some appropriate way) such that, in the future, you can use it with one or more other devices or services without having to authenticate every time.

This is what happens when you get in your car and it connects to your phone automatically, because—from the phone's perspective—the car is a trusted device.

Just as with Application-Specific Passwords, which you can revoke at any time, there's nearly always a way to *untrust* a previously trusted device if it should fall into the wrong hands. The steps vary by device, but usually they involve signing in to a website, finding the security settings, and removing the no-longer-trusted device from a list. Thereafter, it won't automatically connect to your other devices or receive SMS codes or whatever the trusted function was.

Physical Keys

Earlier I mentioned a number of objects besides cell phones and secure tokens that could qualify as "what you have" factors. A popular example—which, conveniently, doesn't require any exotic hardware to use—is a USB key. (Variants are available that use Lightning connectors for iPhones and iPads with Lightning ports.) The general idea is this: On your computer or other device, enter your username and password as usual when prompted. But when it comes time for that second factor, you don't need to look up or enter a numeric code. Instead, simply plug in a tiny dongle and press or tap a button (or, if the device is already plugged in, just press the button). If a USB or Lightning port is unavailable or too inconvenient, you may be able to wave a keychain-sized token over a wireless reader—perhaps even one already built into your smartphone or computer.

Behind the scenes, there are a number of technologies that can be used to pull off this feat. Most commonly, you'll see support for U2F (Universal Two-Factor) authentication and/or a newer variant, FIDO2, in software (such as cloud-based password managers), services (such as Dropbox), and physical keys. Many such devices exist, and you can purchase whichever one strikes your fancy—often for the proverbial price of a latte.

For example, as I write this in January 2025, the HyperFIDO Titanium Pro FIDO2 Security Key costs less than $20 at Amazon.com. YubiKey products by Yubico are more expensive, but some are also more versatile, supporting numerous additional authentication protocols besides U2F. Although most U2F/FIDO2 tokens use USB or Lightning, some use NFC (Near Field Communication) chips instead or in addition, offering wireless connections with compatible devices.

Note: The YubiKey 5Ci features both USB-C and Lightning connectors, which should be suitable for any recent Mac, iPhone, or iPad, as well as most recent PCs and Android devices.

Smart cards (with embedded chips, similar to credit and debit cards) are another example of physical keys that can be used for 2FA; these need a specialized piece of hardware to serve as a reader. Physical keys with other form factors and technologies (and designs that integrate security tokens into other devices) also pop up from time to time.

Note: For devices you may use *in place of* entering passwords—rather than as a second factor—refer back to Authenticator Devices.

Two-Factor Pros and Cons

Now you should have a pretty good idea of what two-factor authentication options exist and how they work. But the question is: given the choice, should you use two-factor authentication (or two-step verification)? The answer isn't as straightforward as you might think.

Here is a complete list of pros to using two-factor authentication:

- **Greater security.** Seriously, that's the whole point—there's nothing more to say. How much greater the security is depends on which two factors are involved, but on average, it's safe to say that you massively decrease your risks with each added factor.

What about the cons? Try these:

- **Human fragility:** You might have to carry an object with you (a token, USB key, cell phone, etc.) and if you lose or break that object, you won't be able to log in—at least, not without considerable hassle. Similarly, if you use a biometric factor (voice recognition or a fingerprint, say), an injury or illness can trip you up.
- **Technological fragility:** The additional hardware used for some factors (smart-card readers, fingerprint scanners, and so on) can malfunction or be thrown off by environmental anomalies—again, preventing you from logging in.
- **Inconvenience:** Two-factor authentication almost always involves something extra for you to do. Even if it's a simple tap, swipe, scan, or whatever, it forces you to do more work every time you log in.

Note: Some forms of biometrics try to overcome the inconvenience problem in clever ways. For example, a system might watch the way you type to discover distinctive patterns in the speed with which you enter certain combinations of characters. That way, without doing anything other than typing your password, you add a "what you are" factor.

Does the pro outweigh the cons? The answer is different for everyone. Having to enter extra codes to log in is a drag; some users have found it sufficiently aggravating that they've given up and gone back to standard authentication. On the other hand, depending on how the two-factor or two-step system is designed, the extra effort may be minimal—and the additional security can be a lifesaver. And, two-factor authentication or two-step verification is sometimes mandatory, regardless of your feelings about it.

My suggestion is to try two-factor authentication in situations where it's optional—as long as there's a path to return to one-factor authentication. Use it for a while and see if you get used to the extra steps. If you feel you can live with them, bask in your extra security. If not, go back to the old way.

I explain how to use 2FA (or, more frequently, 2SV) for several common services in Appendix A: Use Two-Factor Authentication.

Authentication, Unlocking, and Decryption

Throughout this book, I've used the term *authenticate*—literally, "prove your identity"—to mean "log in to a device or service with your username and password." That is, when you authenticate to a computer or website, you're proving that you are who you say you are. You can't authenticate with a service that doesn't already know about you; you must have an account first. The whole notion of authentication assumes that the party you're authenticating with needs to know *who you are*.

All that may seem obvious, but it gets trickier when you consider cases where you must supply a password but *not* a username, email address, or other unique identifier. For example, when you unlock your smartphone, you need only a passcode—not a username. Likewise, when you unlock a password manager by entering its master password (and thereby decrypt its contents), you aren't authenticating at all, even though you're entering a password.

I bring this up now, in the context of multi-factor authentication, because there can be no multi-factor authentication if there's no authentication in the first place! MFA does not apply in situations where you supply only a password, because you're not logging in to an account that's tied to your identity.

Some password managers *do* offer a 2FA (or 2SV) option. But these usually store your data in a unique account in the cloud, so you are in fact authenticating in the process of unlocking your password vault. Although I can imagine situations in which a second form of identification would be helpful as a security measure even when no authentication is involved, providing such a system would involve significant technical challenges and risks. So before asking whether there's a 2FA option, make sure there's an A in the first place!

Authenticating with Another Site's Credentials

A large number of websites and web-connected apps offer an alternative to entering a username and password. On a login page, you may see "Sign in with Apple," "Sign In with Google," or "Sign In with Facebook" buttons, or similar controls referencing other services with which you may have an account. Sometimes you have the option to use one of these services instead of creating a new password for the site you're visiting, while in other cases it's mandatory—you can log in *only* with an account from another service.

This type of cross-site login usually uses a method called OAuth (for "open authentication"). Let me give you an example of how it works.

You go to the YouFace site and see a Sign In with Google button. Click it, and you'll go to a page on the Google website. If you aren't already logged in to Google on your current device, you'll be asked for your Google credentials. Then you'll see a Sign In button and a list of activities YouFace may and may not perform once you grant permission. For example, YouFace may want to see the personal info Google has about you, but not update it. If you agree by clicking Sign In, Google passes a small chunk of data called a *token* to YouFace, which essentially says: "The person who clicked that button is the owner of the Google account in question."

From that point on, YouFace can access whichever parts of your Google data it requested, but your Google password itself is never transmitted to YouFace. If you later decide you don't want YouFace to access your Google account anymore, you'll have to log in to your account on the Google site, dig through your settings, and revoke access for that service or app.

Sites that use OAuth typically emphasize two key points. First, logging in this way is easier because you don't have to create or memorize a password—usually it's just one click, and you can keep accessing the service indefinitely. Second, it's more secure because you don't have a

password sitting on the site's servers waiting to be hacked. (The token is kept on your device, not the server.) All that is true, as far as it goes.

However, there are some hidden gotchas. For one thing, you don't have a line-item veto (as it were). You can't say, for example, "Sure, you can see my name but not my birthday." You can only agree globally to the lists of activities the site says it does and doesn't want to do. And in some cases, you may not be able to tell exactly what sorts of control you're giving away—for example, if you use your Google account for authorization, it may not be obvious that another site could read all the messages in your Gmail account! Therefore, you must trust the sites you use this way to protect your data and behave with integrity. Unfortunately, many sites take the opportunity to mine your data from Facebook or LinkedIn or wherever in order to display targeted ads or do even less-savory things—such as posting things in your name that you may not agree with.

Furthermore, the burden is on you to remember which sites or apps you authorized with which of your accounts. If you decide that you want to stop using YouFace a year from now and block its access to your data, you have to remember which set of credentials you used (was it Apple, Facebook, or Google?), go to that site, and figure out how to revoke authorization.

And, if you decide to stop using a service and you close your account, any *other* sites that relied on that account to provide authentication will no longer be able to confirm that you're you when you next try to log in on a new device.

Furthermore, if you discover or suspect that a password has been compromised, merely changing that password doesn't log you out of apps or devices you've previously authorized using OAuth. So, if an attacker logs in with your stolen password and then you change it, the attacker isn't kicked out automatically. The solution? In addition to changing your password, if you used a third-party site such as Apple, Facebook, Google, or LinkedIn to authorize other sites, log in to the account whose password you just changed and revoke authorization

for all apps and devices listed. You'll have to authorize each one again the next time you use it.

For all these reasons, I personally avoid using this sort of third-party authentication when I have the choice. Creating a strong, random password with my password manager takes just a click or two, and entering it when I need to access the site later is just as easy. OAuth isn't inherently bad, but I like keeping tighter reins on my private data.

Apply Joe's Password Strategy

In my earlier book on passwords, I distinguished between "identity" and "security" passwords and outlined elaborate techniques to determine how strong a given password needed to be and create different kinds of passwords depending on context. I now advocate a single approach that's simpler and safer, and that covers the vast majority of cases.

My strategy—and yes, this is what I do myself—has three main points:

- Figure Out Which Passwords You Must Memorize—if you do it right, the number of these passwords will likely be in the low single digits.
- Create Strong but Memorable Passwords for just those few. The passwords should be strong enough to defeat all but the most determined hacker yet easy to recall and type.
- Use a Password Manager for Everything Else. Your remaining passwords will be long, complex, and random. You'll have no idea what they are, but you won't have to, because you'll almost always be able to enter them with an automated tool.

You'll also have to deal with irritating security questions from time to time, as well as other odd exceptions and surprises. I cover all that in this chapter as well.

Note: Although, strictly speaking, it's not part of a *password* strategy, if and when you come to a site that permits you to create a passkey instead of (or in addition to) a password, you should absolutely do so! See Authenticate Without Passwords for more details.

Figure Out Which Passwords You Must Memorize

First, the bad news: you *must* memorize at least a few passwords, and those few have to be both long and strong.

But the good news is that for most people, with careful planning, the number of passwords that must be stored in the brain is very small. For me, the number is four. You might have only one or two, or you might have nine or ten—but if your number gets much beyond a dozen, *you're doing it wrong*. Whatever the number is, I'll refer to this short list as your Very Important Passwords, or VIPs.

Which passwords belong on the must-memorize VIP list? Only those passwords that you need often and can't easily enter using a password manager app (which I discuss two steps ahead). For example, here are my four:

- **The master password for my password manager:** A password manager lets you use a single *master password* to unlock all your other stored passwords. I use that key constantly and I can't very well keep it in my password manager, so I have it memorized.
- **My computer's login password:** Everything on my computer is encrypted, so I can't turn it on or even wake it up from sleep (much less run an app such as a password manager) without entering the login password for my main user account.
- **My phone's passcode:** I follow my own advice to make my phone's passcode longer and more complex than a short, numeric code, so that has to be memorized too.
- **My Apple Account password:** My Apple Account (called an Apple ID prior to late 2024) can get me into all sorts of services—iCloud on my Mac, PC, iPhone, iPad, and Apple TV; my iTunes Store and Mac App Store accounts; Game Center; my Apple developer account; and so on. I enter it so often that it was well worth memorizing. (I have more than one Apple Account, but I use one

much more than the others.) See Devices Without Full Keyboards, later in this chapter, for additional advice on passwords—such as an Apple Account password—that must be entered frequently on a tiny virtual keyboard.

Those four passwords are the only ones I feel I have to keep in my head. My other 1,000+ passwords (really!) are either web logins that my password manager can enter for me or passwords I use so seldom that I don't mind looking them up and then copying and pasting (or retyping, as the case may be) when the occasion arises.

Your VIP list might be different from mine—and it might be longer or shorter. But the one item that's non-negotiable is the master password for your password manager, because a large portion of this strategy depends on it.

The best way to figure out which passwords should make your VIP list is to keep a tally of the passwords you enter manually on any given day *outside a web browser*. Anything that shows up on that list more than a few times is a good candidate. But don't include passwords you enter on webpages, because a password manager can handle those for you. Your list might include a school or corporate login password, a Wi-Fi network password, passwords used in individual iPhone/iPad apps, or the password for any account you have to access on a device where you can't install your own password manager.

Once you have your list of VIPs, you're ready to…

Create Strong but Memorable Passwords

Maybe one or more of the passwords on your VIP list already has extremely high entropy (refer back to All About Entropy). If so, you need only commit it to memory—more on that in a moment. For each of the rest, your task now is to create a *new* password that's adequately strong, memorize it, and change your existing password to the new one. As you do, be sure to write these passwords down (yes, on paper).

These are the ones you can't afford to lose or forget, and as I explain in Keep Your Passwords Secure, having a carefully hidden paper copy of your passwords might actually *increase* your security.

> **Note:** Remember, you're not doing this for all your passwords. Do it only for those few you identified a moment ago as being essential to memorize.

How long must this password be, and how can you remember it?

Those questions are two sides of the same coin. As I explained earlier, the important thing is to make the password strong enough—to give it enough entropy—to resist a brute-force attack. You can derive that strength from randomness, length, character-set size, or a combination of these.

If you want the fewest possible characters, the characters must be random and varied; if you want a password that's a breeze to type because it's all lowercase words, it must be much longer to have equivalent strength. You're the one who has to type these passwords all the time, and you know how your memory works better than I do. So I'll leave that decision to you.

Here's how a few of your options compare:

- **Random:** Mathematically speaking, random passwords using all available character types pack the most entropy into the fewest number of characters. So, if you have a good memory and want to save yourself a considerable amount of typing, use the password generator in your favorite password manager (see the next topic, Use a Password Manager for Everything Else) to create a random password that includes upper- and lowercase letters, digits, and punctuation—and is at least 12 characters long.
- **Sentence/pronounceable:** Passwords created using the first letter of each word from a sentence or a password generator's "pronounceable" option may appear random, but they have lower entropy because they do contain recognizable patterns. You can use

such a password if you prefer, but make it longer—16 characters or more (assuming they include either digits or punctuation).

- **Lowercase words:** If you go for the simplest possible password to type and remember—a series of ordinary, lowercase English words—then you need many more characters to reach a comparable level of entropy. If you choose something in the `correct horse battery staple` vein, make sure it consists of at least *five* words—all of which are randomly chosen—and has at least 32 characters. (A four-word, 28-character phrase like `correct horse battery staple` arguably has insufficient entropy to resist modern cracking methods.)

  One good but time-consuming way to create a sufficiently complex passphrase of this type is to use the Diceware technique I mentioned earlier, which requires actual dice, a downloadable word list, paper, and a few minutes per password. (Alternatively, you can use a special set of physical dice called DiceKeys, but it requires a $25 investment in a gadget you may use only once.) There's also an online password generator based on the XKCD method, and 1Password's Memorable option, which is like Diceware but more secure because it's based on a much longer word list—and far quicker to use. (There's even a free version on the web that you can use without subscribing to 1Password itself.) You can also try a new method by John Clements called Molis Hai, which creates long, high-entropy passwords that are easier to remember because of their resemblance to English words.

Note: Appealing though this last route may be, remember that many sites and services enforce rules such as "must contain a digit," "must have at least one uppercase letter," and "may not contain spaces." So you must be sure each password is appropriate to where it's used; check a destination's requirements before creating a password.

Now that you have your short list of Very Important Passwords, it's time to memorize them. This will require actual effort, even if you're using a sentence or plain English words (was that `correct battery horse`?—no, wait, `correct staple battery`...). Sorry about that. But you've successfully memorized plenty of other things—your phone

number, your address, your Social Security number—and I have the utmost confidence that you can add a few more items to that list.

For some people, rote repetition is the way to go. With concentration, it shouldn't take more than 15 minutes to memorize a password. Or, try writing or typing the password a hundred times in a row. ("I will not chew gum in school. I will not chew gum in school…") Or, break it down into chunks of three or four characters and memorize those one at a time. Or, turn it into a game to see how many sentences or stories you can come up with to represent your password. Think of it as a small investment of time for long-term savings.

Once you think you have a password down, put it out of your mind, distract yourself with something else, and then try to remember it an hour later. Try again before bedtime and when you wake up in the morning. You should have your entire VIP list down in a day.

What About the NIST Guidelines?

In June 2017, the National Institute of Standards and Technology (NIST) published new Digital Identity Guidelines—including, at long last, updated advice on choosing passwords (the previous guidance was truly awful). The document was updated in March 2020. This advice is geared toward federal agencies implementing password rules, not to individual users. Nevertheless, since it describes what the government considers best practices, it's worth taking seriously.

Most interestingly (see the guidelines' Appendix A—Strength of Memorized Secrets), the NIST no longer recommends that federal agencies mandate complex passwords, and states that *length* should be the primary consideration rather than randomness or character set. The main reasoning is that complexity makes passwords harder to remember and tempts people to do the minimum necessary to conform to the rules (that's how we get awful passwords like `pAs$word1`), thus paradoxically leading to weaker passwords.

That advice is fine as far as it goes, and it meshes well with my own advice about your VIPs. But bear in mind that the NIST guidelines still assume people will be memorizing their passwords. If you use a password manager to create, store, and enter your password, complexity is no longer a problem. So there's nothing wrong with choosing the path of greater complexity to increase entropy if you let an app do it for you.

Use a Password Manager for Everything Else

With your VIP list down cold, it's time to turn your attention to your other passwords. Virtually every other password in your life (I'll mention a few exceptions ahead) can be handled the same way: you'll use a piece of software to create it, store it, and then enter it for you whenever you need it.

I refer to this type of software as a password manager, but you may hear other names, such as password vault or keychain. Password managers wrap up all your individual usernames and passwords—and

sometimes other important data too—in a securely encrypted file that you can unlock with a single master password.

Since you have only one password to remember, the others can be long and complex. You won't need to see them. You won't even know what they are. They'll be created for you randomly by a feature called a password generator. Then they'll be stored automatically, and—at least in the better password managers—entered into web forms for you with the flick of a finger.

You'll install a version of this software on your Mac or PC as well as on your smartphone or tablet so you can keep your passwords in sync among devices. You may even choose a manager that lets you access your passwords securely from any web browser, or keep a copy of the app and its encrypted data on a flash drive in your pocket. That way, as long as you are in the vicinity of a computing device of some kind, you'll have a way to get to all your passwords.

No Smartphone or Tablet?

I'm making the bold assumption that most people reading this book have a smartphone or tablet that is always handy and can run a password manager.

If you don't have such a device and you frequently need to use passwords when you're not at the computer where your password manager is installed, this system will be less convenient. Keeping your password manager and its data on a USB flash drive may be a solution. Alternatively, you might use a web-based password manager such as the one offered by 1Password. For further ideas, see Appendix B: Help Your Uncle with His Passwords.

It doesn't matter to me which password manager you choose. There are tons of them out there; I mention nine popular examples in the next chapter, Pick a Password Manager. The important thing is that the app meets your personal needs and works with all the devices and operating systems you care about.

Once you've chosen a password manager, install it on all your devices, set it up with a strong master password, and if it includes an automatic sync feature, turn that on. Then follow these steps:

1. When a site or service asks you to create a new password, use your password manager's password generator. It'll likely have lots of options, but my recommendation is to let it create the longest and most complex random password that the site will accept. Maybe one site restricts you to 10 characters and another doesn't accept punctuation, while a third lets you create a 64-character password containing any imaginable symbol.

   Whatever the case, let the site's maximum be your guide—it's no more difficult or time-consuming for a password manager to hand you a 64-character password than one with 10 characters, so there's nothing to be gained by making some passwords shorter or simpler, even if you feel they're of trivial importance.

   As you encounter sites and services for which you previously created passwords, type or paste them in manually, but then immediately capture your credentials with your password manager so they'll be ready for you the next time you need them.

2. When you visit a site for which you've previously stored credentials, your password manager should be able to fill them in for you. Depending on which app you use and on which platform, this might happen automatically; it might require a click, a tap, or a keystroke; or you might have to copy and paste. But in any case, you should rarely if ever have to retype a password.

3. When time permits, follow the steps in Audit Your Passwords to replace any old, weak passwords with new, strong passwords.

Along with these steps, I strongly recommend that you follow a few other key practices:

- Always use long, unique, randomly generated passwords, even if the password merely serves to identify you and doesn't protect any valuable data or resources. Read the sidebar Why Use Secure

Passwords for Throwaway Accounts? for a longer explanation of why I recommend this.

- Do *not* enter a password hint when a system gives you that opportunity. No matter how innocent or obscure you may think your hint is, it can help an attacker who's trying to guess your password. Password managers eliminate the need for hints! (Some password managers prompt you to enter a hint for your master password, however. Personally, I'd feel better writing down that master password in a safe place and leaving the hint blank or filling in "No.")

- If there's any possibility of someone else using your computer without your permission, avoid selecting those "remember me" or "keep me signed in" checkboxes on webpages. Such pages use cookies stored on your computer to identify you, eliminating the need to log in manually the next time you visit the site. Since your password manager can autofill your credentials, there's no point in leaving any such security holes that someone else could exploit. For even greater security, delete the cookies already in your browser.

- Related to the last point, most browsers have their own mechanisms for storing and autofilling passwords. When you see a "save this password" dialog box after logging in to a website, you're being prompted to store the password in your browser or in a systemwide keychain. Again, I suggest that you stop agreeing to these prompts, turn off the feature in your browser, and delete any passwords it has already stored. Otherwise, the security you're gaining by using a password manager may be nullified by having a less-secure way to access the same information—and one that likely can't sync across all your browsers, devices, and operating systems.

  A rather terrifying example of how problematic browsers' built-in password managers can be appeared in the news in December 2017. (See No boundaries for user identities: Web trackers exploit browser login managers.) It appears that scripts found on numerous websites were taking advantage of a flaw in the design of built-in password managers to extract email addresses (presumably for the purposes of user tracking and profiling) without the users' knowl-

edge. Basically, these scripts create invisible login forms, let the browsers' password managers fill them in automatically, and capture the information they contain without you, the user, ever seeing anything! Issues like this put the lie to claims that browsers' built-in password managers are inherently safer than third-party password manager apps that use browser extensions. (Password managers that use browser extensions are also superior in the sense that they let you use the same set of passwords across multiple browsers.)

In any case, 1Password's current design makes it immune to this type of problem, though it could potentially occur with certain other apps if they're configured to autofill and autosubmit credentials as soon as a page loads—something I recommend against (see the sidebar Four Autofill/Autosubmit Models).

Note: I'm not going to detail the steps for deleting cookies or turning off password storage and autofill, because they're different for every browser on every platform. Look through your browser's preferences, and if you can't find these features, check the Help menu.

Why Use Secure Passwords for Throwaway Accounts?

Lots of people have asked me why they can't just use simple, insecure passwords for sites that store no personal data. "I don't care if I get hacked, because there's nothing of value there," they say. "Why bother creating complex passwords for those sites?" And, if the security of those passwords is truly irrelevant, why not just go a step further and use the same simple password for all those throwaway accounts? "Sure, I'll use unique and strong passwords for accounts that are truly important," people say, "but *only* for those."

I understand that reasoning; in fact, I used to think the same way myself. On the surface, it's not illogical. But if you think about it more carefully, it makes less and less sense:

- If you use a password manager to autofill your passwords, as I recommend, then it's no harder to fill in a random, 50-character, super-secure password than the word `horses`. In fact, it's easier! It might require just one click, tap, or keyboard shortcut to enter your random password, whereas typing even a simple word like `horses` requires six key presses (plus Return or a click on the Submit button). A simple password actually requires *more* effort!
- When choosing a password for a new site, the same logic applies. A password manager can generate and fill in a random password for you with a couple of clicks or keystrokes—it's no more difficult or time-consuming than typing your go-to password.
- A service that stores no personal data about you now might expand or change in the future, such that it becomes a bigger security concern. And even sites that apparently store no personal data can track your behavior and build up a profile on you—information that you wouldn't want an attacker to be able to use.
- If you reuse the same insecure password on many sites and one of them is hacked, the attacker could log in as you on all your "throwaway" sites. So, what you may perceive as nothing more than a minor irritation on one site could turn into a much larger risk—take the previous bullet point and multiply it by the number of sites that use the same password.

In short, you should use strong passwords everywhere because it's easier and safer—and as long as you're using a password manager, there's no reason not to make every password equally strong.

A Question of Trust

I've had spirited debates with people who insist I'm crazy for trusting a password manager with my sensitive data. How do I know it's safe? What if it has a secret back door and the developer (or the government) is quietly collecting all my passwords? What if it has a security flaw that makes it less safe than it appears? What if (insert your favorite anxiety here)?

Although I'm well aware of many threats to personal privacy, I don't worry about my password manager—not even a tiny bit. Here's why:

- The encryption algorithms used by most (if not all) password managers have undergone lengthy, rigorous, public scrutiny by top cryptographers, even if the password manager's code itself is not open source (see How Secure Is Your Password Manager?).
- It is in the economic best interest of a developer to make your password manager as secure as possible. If a flaw (intentional or otherwise) were discovered and publicized, it could put the developer out of business in the blink of an eye.
- In the case of my favorite password manager (1Password), I've met the company's founders personally and had many discussions with them—enough to convince me that they're trustworthy and have their users' best interests at heart.
- Because both people and machines are imperfect, trusting anyone or anything could conceivably get you into trouble. And yet, we all must trust people and things every day, just to function normally. My locksmith, accountant, or doctor *could* be secretly trying to rip me off or harm me, but I have no reason to think they are, and I'm better off trusting them than trying to repair my own locks, do my own taxes, or perform surgery on myself. Likewise, my car *could* have hidden tracking device and my desk lamp *could* contain a hidden camera, but I have no reason to suspect they do, and that sort of paranoia isn't helpful.
- If I chose not to use a password manager, the alternatives (memorizing hundreds of strong passwords, using fewer and/or weaker passwords, or manually typing in every password from a written list) are themselves incredibly problematic; I'd be trading one problem for another.

So, my professional opinion: you have better things to worry about.

Handle Security Questions

Earlier, in Understanding Security Questions and Reset Procedures, I talked about the security questions you may be prompted to fill in, and the various ways those are used. For all practical purposes, you should treat these as alternative passwords for each account—which means taking the same care in generating them and storing them.

When asked to supply the answer to a security question, there's one rule: *Lie*. Never, ever type in your mother's real maiden name, your third-favorite dessert in kindergarten, or your best friend's pet's name. The facts related to typical security questions are too easy for someone else to discover, and they weaken rather than strengthen your security.

Give answers that are not merely untrue but irrelevant. For example:

- Smash a couple of unrelated words together. My favorite sports team? `brandishedcontumely`
- Use your favorite password generator to create something random. My favorite teacher? `MEk8^RL3{Xvu`
- Vent. The phone number I remember most from my childhood? `I-refuse-to-engage-in-this-nonsense`

A qualification I want to mention is that in some cases, you may have to answer security questions over the phone. If you do, you'll find it easier to say (and easier for the person on the other end to understand) if it's made of English words.

Store the question(s) and corresponding answer(s) in your password manager, preferably in the same record that holds the username and password for the service in question. This is essential because you'll need to regurgitate them later, and you'll no sooner remember these answers than you will the rest of your random passwords.

Unfortunately, your password manager may be unable to fill in the answers to your security questions, since a site may randomly choose any of several questions—and may do so only on occasion. I say more about this in Sites That Thwart Password Managers.

Manage Email Options

Another point I raised in Understanding Security Questions and Reset Procedures is that many sites send email to whichever address you've given them if you ever forget or need to reset your password—so that email account and its password should be given special protection. In many cases, you can't distinguish between the email address the site uses for day-to-day communication with you and the address used for password resets, but sometimes you're asked explicitly for a secondary email address to be used for security purposes.

The safest way to handle email addresses used mainly or solely for password resets is to set up a new email account—perhaps using Gmail or another free service. I don't mean an alias pointing to your existing inbox, but a separate account with its own password and an address that you never share publicly. When a website or service asks for an address to use to verify your account or reset your password, use that new address. Keep its credentials handy so you can log in and check messages when needed, but don't add it to your regular email client, where someone might gain access without needing a strong password.

If you choose instead to use your regular email account, take extra precautions with it:

- Make sure your email password is strong, and that you use SSL to access the account securely on all your devices. If your email provider doesn't offer SSL and you can't switch to a better provider, at least make sure to use an encrypted authentication method—such as Kerberos version 5 (GSSAPI), NTLM, CRAM-MD5, or Apple Token—so your password can't be sniffed while in transit.
- If feasible, set all your devices to lock when not in use and require a strong password to unlock them.
- If your account supports it—and most do these days—use two-factor authentication or two-step verification. See Appendix A: Use Two-Factor Authentication for details.

Deal with Exceptions and Surprises

So far, the strategy I've outlined works well for most websites that use straightforward username-and-password authentication. But some sites, services, and resources that require passwords have unusual attributes that throw a wrench in the works. Let's look at how to handle some of these.

Places Your Password Manager Can't Reach

Most password managers can fill in forms on webpages, but you may have to enter passwords in other spots on a computer or mobile device that are inaccessible to your automated tools, such as:

- Systemwide preferences or settings
- Websites that use basic access authentication (see Basic access authentication at Wikipedia) or digest access authentication (see Digest access authentication) instead of, or in addition to, form-based password entry
- Local network file servers, which you connect to outside a web browser
- Apps that lack hooks into third-party password managers such as 1Password, or even intentionally block such access
- Devices on which you can't or don't want to install a password manager

In most of these cases, the best (albeit clunky) solution is to copy the password from the password manager and paste it into the desired location—or, if that's impossible, view it in the password manager and then type it manually. It's not fun, but in my experience these situations occur seldom enough that copy-and-paste is not a hardship.

Tip: In macOS, the systemwide Keychain (see Apple's Password Tools) can store and fill in some of these problematic scenarios and more—including passwords for websites that use basic or digest access authentication, Wi-Fi networks, local network file servers, and even encrypted disk images.

Limits on Password Length or Character Set

Suppose you've set up your password manager so that, by default, it generates 32-character random passwords containing all kinds of characters. Then you go to a website, click the button to create and fill in a new password, and get an error message, such as "maximum password length: 8 characters" or "invalid character." In these cases, you'll have to adjust your password manager's settings and try again, and the end result may be a password that's weaker than you prefer.

Note: 1Password's password generator includes a Smart Password option, which attempts to discern the password requirements of the site you're visiting and choose a password that fits those parameters.

I'm sorry to say that these sorts of restrictions are all too common. There's no excuse for imposing such limitations—the amount of effort and processing power needed to handle longer passwords and larger character sets is almost trivial. All the same, there's nothing you can do except adhere to the requirements and write a polite note to the system administrator complaining about this insecure practice. (In your note, be sure to point out that you aren't a mind reader and can't know what the site's password rules are unless they're spelled out on the page.)

The reverse of this problem is also common. If you want to use a simple-but-strongish password like `correct horse battery staple`, you'll have no luck with sites such as Apple's, which mandates that all passwords include a number, an uppercase letter, and a lowercase letter—and no spaces.

A much worse problem, which I've seen more than a few times, is when a site accepts a password when you first sign up, but then rejects the

same password later when you try to log in! The most common reason for this is that the code used to process passwords on logins chokes on special characters (such as quotation marks) that aren't checked for when you select a password initially. Once again, there's nothing for it but to change your password and then complain—whatever restrictions a site has, they should at least be internally consistent.

Note: Another common requirement you can't control is a site's demand that you change your password every so often (see the sidebar Should You Change Your Passwords Regularly?). Aren't you glad you have a random password generator handy?

Devices Without Full Keyboards

It's not particularly hard to enter the password `MEk8^RL3{Xvu` on a Mac or PC. But typing that on an iPhone keyboard? Ick. It takes no fewer than six keyboard changes, not counting the characters themselves (or the Shift key for the uppercase letters). And how about entering that password on a streaming video device connected to your TV with nothing more than a D-pad remote control? I hope you weren't in a hurry to watch a show, because it'll take a while. And you'd better enter that password carefully—if you make a mistake you'll have to start over from scratch!

Of course, you'll have a password manager app on your iPhone or other mobile device, but every once in a while you may have to enter a password in an environment in which not only autofill but even copy-and-paste is disallowed. Unfortunately, you may not be able to predict which passwords will require that treatment. But if you know you'll be entering a password on a device without a full keyboard, you'll have a vastly easier time doing so if you limit the number of times you must switch the virtual keyboard layout. (This is another argument for `correct horse battery staple`-type passwords, which require no keyboard switches, although unfortunately in many cases, you're obligated to include at least one uppercase character and one digit.)

Tip: If you have an Apple TV device and an iPhone or iPad, the Apple TV Remote feature in Control Center lets you use your device's keyboard to type (or even copy and paste) passwords on your Apple TV—a huge step forward in usability. Apple TV devices beyond the first generation also support Bluetooth keyboards. Even better, Apple TV HD and 4K models let you dictate a password, one character at a time, using the Siri Remote.

Sites That Thwart Password Managers

The last category of exceptions and surprises I want to mention is websites that—thanks to measures intended to prevent automated attacks or improve security for the unenlightened masses without good password strategies—make life more difficult for us evolved beings who have excellent passwords stored in equally excellent password managers. The most common offenders are financial institutions.

Here are some of the problems I've encountered:

- **Multi-step logins:** Password managers are designed for the standard situation in which a single login page displays both username and password fields. But some sites, including Apple's icloud.com, separate these into two pages (or even more, adding an email address or other item to be entered), each of which must be submitted separately. Fortunately, most modern password managers can handle these correctly.
- **Rotating security questions:** A special instance of multi-step logins, some sites ask you a security question every time you log in—but a different question each time. Filling these in automatically is almost out of the question, generally forcing you to look up the question of the day, copy the answer, and paste it in.
- **Paste prohibitions:** To make things even more frustrating, some sites prevent you from pasting anything into the password field. This may interfere with your password manager's autofill feature. In a few obnoxious cases, even choosing Edit > Paste (or right-clicking/Control-clicking and choosing Paste from the contextual menu) won't work, forcing you to manually type your password.

- **Grids:** Some banks, such as the one I used while living in France, have ordinary fill-in fields for your account number or username, but then display a grid (**Figure 1**) on which you must laboriously click or tap a numeric password. Because the positions of the numbers are different each time the page is loaded, there's no way for an automated cracking system—or your password manager—to know how to click the right sequence; you must always do it manually.

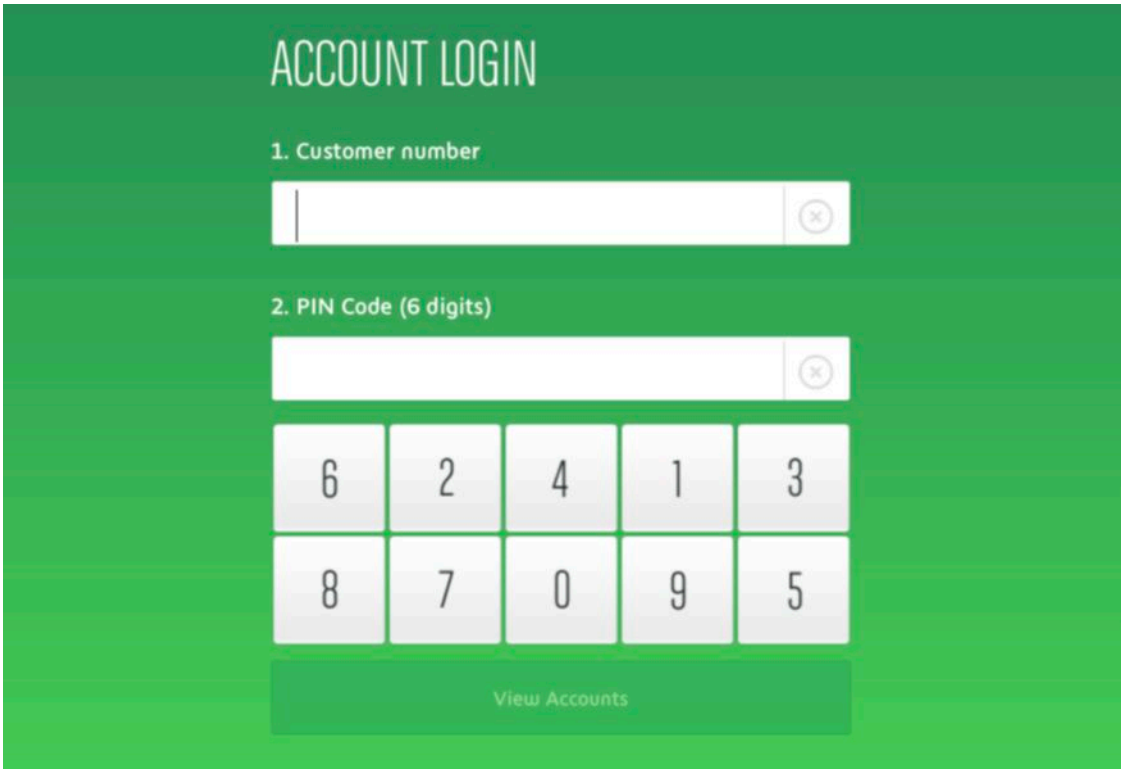


Figure 1: To log in to your account at a certain French bank (and many U.S. banks), you must click an ever-changing sequence of squares on a grid to enter your numeric password.

Other than creating multiple password entries for a single site (which helps with some of these problems), I have no solutions. (Remember how I said I can probably help you solve 98% of your password problems? This is part of the other 2%.) We can only hope that the good people who develop password managers will talk to the equally good people who protect your money and find clever ways to address these problems in the future.

Pick a Password Manager

A quick web search will turn up dozens, perhaps hundreds, of password managers. Because I talk so much about password managers in this book, I want to offer some advice about how to choose one. In this chapter, I introduce you to the capabilities you may find important and then offer a brief overview of nine representative password managers.

If you're already using a password manager and you're happy with it, you can probably skip this chapter (or skim it, to see if anything interesting pops out). But if you're using a password manager that you aren't entirely satisfied with, use this chapter as a way to find something that may be a better fit. Switching password managers can be a hassle (check out the sidebar just ahead for advice) but it's worth the effort if your new manager makes it easier for you to consistently create—and easily access—strong passwords on all your devices.

Remember, this chapter is only a sampling of your options; my point is to acquaint you with the variety of choices out there rather than to push you to use any particular app. I'd much rather you used my least-favorite password manager than none at all!

Tip: If reading about password manager features makes you drowsy and you'd rather pick one and get on with it, allow me to suggest 1Password. It's what I use, and I think you'll like it. If you're more concerned about cost than anything else, use Bitwarden. Dashlane easily makes my Top 3 list too. Also see Joe's Recommendations at the end of this chapter.

Switching Password Managers

Say you have a bunch of data in a password manager, but you start using a new platform that your current manager doesn't support. Or you find out that your current manager has a security problem. Or you're tempted by fancy new features in another manager. In all these cases, the question is: how easily can you transfer your stuff from the old password manager to the new one?

There's no easy or universal answer. Many password managers (including 1Password, Dashlane, and RoboForm) offer import and/or export capabilities, but even then, there's no guarantee that your new password manager will be able to import the particular format that your old one uses (or exports to). Moreover, even if you can find compatible formats, you might still lose data—for example, if you store custom fields or file attachments in 1Password, those pieces of data will go missing when imported into a password manager without comparable features.

I suggest that before you spend any money, you confirm that the new password manager you're considering can import data from your old one (or from a format the old one can export). You may need to search the support site for each app to see its capabilities—or, better yet, download a free trial version if it's available and try it.

A particular pain point is trying to import data from Apple's Keychain format. It can be done—for example, I just tried it with Dashlane—but there is a catch. Because of Keychain's security design, you must individually approve every single login item as you import it (which sometimes means clicking Allow, and other times may mean entering your password). If you have hundreds of items in Keychain, that can be an enormously time-consuming and tedious process.

Features to Look For

Every password manager starts with the same premise: put all your passwords (and, often, other private information) inside this secure storage place, and unlock it as needed with a single master password. Superficially, most password managers even look similar—they're essentially encrypted databases with predefined fields for username, password, URL, and a few other items.

Beyond the basics, however, there are a great many features that can spell the difference between a great password manager and a mediocre one. Not everyone cares about the same features, so you may want to create your own list of priorities as you consider various password managers. To get you started, here are a few key things to look for:

- **Platform support:** As I mentioned, most of the password managers I cover in this book are available for Mac, Windows, iPhone/iPad, and Android. Some support additional platforms as well (such as Apple Watch, available via the iPhone app, and Linux), though you may find that not all features are available on all platforms. In any case, make sure the app you choose works on the platforms you personally care about.
- **Mac and Windows browser extensions:** Any password manager lets you search for a stored web login and click a link to open that URL in your default web browser. But what if you're already in your browser when you realize you need to enter your credentials? Better password managers include browser extensions for macOS and Windows that let you access your credentials within your browser. Usually, though not always, extensions are available for Firefox, Google Chrome, Edge, and Safari (on Mac). Most also support other browsers.

  Some password managers include *all* their functionality in a browser extension, with no companion app. In other cases an app is either mandatory or optional.

Note: The privacy-friendly Epic and Vivaldi browsers, based as they are on Chromium (the foundation of Chrome), support most Chrome browser extensions; the same can be said of Arc, Brave, Opera, Orion, and numerous others. So if a password manager supports Chrome, it should support those, too, even if the developer doesn't explicitly call out support for each browser by name.

- **Mobile features:** The story for mobile platforms is more complex, because mobile browsers generally do not support the same kinds of extensions as their desktop counterparts. Mobile devices are also

more likely than laptop or desktop computers to offer fingerprint and/or face recognition for unlocking your password manager (in lieu of a master password).

When it comes to autofill (about which I say more in the next bullet point), capabilities have steadily improved on both iPhone/iPad and Android. Nowadays, any password manager worth considering supports the systemwide autofill capability on an iPhone or iPad. Similarly, Android 9.0 Pie and later has a systemwide autofill framework that any competent password manager should support. (Android 8.0 Oreo has less-extensive capabilities, while earlier versions of Android relied on clunky solutions like alternative keyboard layouts to autofill usernames and passwords.)

- **Autofill:** Nearly all password managers, on nearly all combinations of platform and browser, have some automated way of filling your credentials into a web form—typically with a keyboard shortcut or a click/tap on a button. (Some password managers can fill in your credentials as soon as a page loads, with no clicking or key presses required.) You may also want the option to choose from among multiple sets of credentials for a given site. (For example, if you have more than one Gmail account or Apple Account, it's nice to be able to choose among them on the fly.)
- **Autosubmit:** Filling in your credentials is one thing; clicking the Submit (or Log In or whatever) button is another. Some password managers optionally do that for you automatically at the same time they autofill your credentials. Others require you to click a button or press Return manually, which may be a tiny bit of extra effort, but also reduces security risks. But there's more than one approach to autofill and autosubmit, as the sidebar ahead explains.

Four Autofill/Autosubmit Models

Depending on your operating system, browser, and password manager, you may have one or more of the following options:

- **Autofill on load.** Some password managers fill in your credentials automatically when the page loads, but you must click or tap Submit, or press Return, to submit the form.
- **Autofill on demand.** After a page loads, you perform an additional step—choose a menu command, click a password in a list, or press a keyboard shortcut, say—to fill in your credentials. Then you either press Return or click a Submit button to submit the form. This is the only behavior in some password managers, and an option in others.
- **Autofill and autosubmit on demand.** After a page loads, you perform an additional step (just as in the previous option) to fill in your credentials *and* submit the form in a single action. This is the default behavior in Safari, and optional in 1Password.
- **Autofill and autosubmit on load.** Immediately when a page loads, the password manager automatically fills in your credentials *and* submits the form, with *no* manual interaction at all. This is the default behavior in Dashlane and NordPass (though you can disable it), for example. This is the easiest option, but also the least secure by far, so I recommend against it. It's essentially the same risk as the "invisible login form" exploit I described earlier in Use a Password Manager for Everything Else.

- **Form capture:** When you fill your credentials into a web form manually or using copy and paste, some password managers can automatically capture that information—adding it to their database as a new login item—when you submit the form. (Usually, they ask whether you want them to save the data, and if so, to give it a name and possibly a category.) Other password managers can capture data entered in a form, but only if you manually use a menu command or keyboard shortcut. Still others require you to enter your credentials in a separate app by hand—a far less convenient option.
- **Password generator:** All the apps I mention here can generate random passwords—an important part of my overall strategy. In

most cases, you can specify settings such as how long the password should be and which type(s) of characters it should include (digits, upper- or lowercase letters, symbols). Sometimes you can request passwords that are pronounceable, "memorable" (usually meaning a Diceware-style word sequence), or have other built-in mnemonic clues—but since most passwords will be filled in for you automatically anyway, I recommend using such options only for your VIP list and keeping the bulk of your passwords purely random.

- **Passkey support:** Support for saving and authenticating with passkeys (using methods analogous to those for capturing and submitting passwords), and then syncing those passkeys across devices, is quickly becoming a must-have feature in third-party password managers. With this feature, you'll be able to use passkeys almost anywhere you need them, even if your favorite browser or operating system lacks sufficient native support for them.
- **Data storage:** Some password managers store your data locally on each device and (optionally) use a cloud service only for syncing that data with your other devices. Others store your data in the cloud, meaning it's accessible on any device with the right software, an internet connection, and your credentials. Both approaches have their merits. Because of strong encryption, I don't think there's a significant privacy benefit to avoiding cloud storage altogether. However, whether or not a password manager uses cloud storage, I feel most comfortable with apps that at least give me a local copy of all my data, so that I'm not entirely dependent on a cloud service.
- **Syncing:** Some apps sync with their counterparts on other platforms using Dropbox, iCloud, or another cloud provider; others rely on their own proprietary cloud service. A few offer direct syncing via Wi-Fi, which may be either a benefit (more secure) or a detriment (more cumbersome), depending on your point of view.
- **Sharing:** If you need to share certain logins securely with other people (such as coworkers or family members), some password managers let you do so. And a few go even further by letting you set up an emergency contact that will gain access to your passwords

only if you die or become incapacitated or unreachable (see Prepare an Emergency Password Plan, later).

- **2FA support:** For password managers that store their data in the cloud, the option to use two-factor authentication (2FA) gives you an extra layer of security—although most of the apps that claim to have 2FA really use two-step verification (2SV) instead; I discuss the difference in One-Time Passwords. (If a password manager doesn't store its data in the cloud, there's no authentication to be done in the first place, so 2FA/2SV support isn't applicable—see the sidebar Authentication, Unlocking, and Decryption.) One way to implement 2FA, which I explained in Physical Keys, is to use a special USB key along with a password to authenticate when you log in to an account. 1Password and Dashlane both support YubiKey USB keys for 2FA.
- **One-time password viewer:** I've mentioned how 2FA and 2SV systems often rely on time-based one-time passwords (TOTPs) generated by standalone apps such as Google Authenticator and Authy. Wouldn't it be nice if your password manager also functioned as a TOTP viewer, so you don't need yet another app? 1Password, Dashlane, Enpass, and the premium versions of Bitwarden and Proton Pass are among the apps that can do this.
- **Security audit features:** As I explain later in Audit Your Passwords, it's important to be able to check on the robustness and security of your passwords, but doing so manually is a pain. Many password managers have built-in tools to help with this process. They typically fall into two classes, which I refer to as "passive" and "active." (The best password managers offer both.)

  Passive security auditing simply flags items in your vault that meet (or fail to meet) some predetermined criteria, such as weak or reused passwords, or those that haven't been changed in a while. Active auditing, on the other hand, is dynamic—typically, the app periodically downloads lists of compromised websites, checks for your username in lists of leaked credentials, and checks for other conditions affecting your security that change over time.

- **Pricing:** It's worth considering how much your password manager is going to cost you—not just today, but over time. Some are free; some require one-time payments or ongoing subscriptions; some use a *freemium* model in which the basic version is free but you pay for premium features. And, among the freemium password managers, some ask for only a one-time payment to unlock those premium features, but more commonly you're charged monthly or yearly. Although I list current prices, be aware that prices change frequently. Check current prices before making any purchasing decisions.

Note: For the purpose of this book, I'm primarily concerned with password-related features. But most of these apps can also store secure notes (and sometimes other files too), fill in contact data and credit card numbers, and help you keep track of other personal data (frequent flyer numbers, software licenses, and so on).

How Secure Is Your Password Manager?

Password managers always encrypt your data; they'd be pointless otherwise. Developers throw around terms like "128-bit AES" and "256-bit Blowfish" by way of touting how secure their encryption methods are, but it's nearly impossible for a typical user who isn't a cryptographer or a computer scientist to validate security claims or to compare one method against another. You pretty much have to take the developer's word that whatever method of encryption a password manager uses is adequate—and in virtually all cases, it is. (For more on this topic, see the sidebar A Question of Trust.)

As long as your password data file itself is strongly encrypted, your data is safe even if it's transmitted over a public Wi-Fi connection or stored in an unencrypted destination such as Dropbox.

But—and this is a big "but"—regardless of the encryption algorithm or strength, your password manager is only as secure as the master password you've chosen. Take the utmost care in choosing and safeguarding a master password, and set your password manager to auto-lock after a brief period of inactivity so that no one else has the opportunity to look through your unlocked data.

Example Password Managers

With those features in mind, let me offer brief descriptions of a handful of password managers to illustrate what I think they do right and wrong. This is not by any means a comprehensive list, nor do I think every app listed here is equally good.

Every app I list below features:

- Native versions—which may mean apps and/or browser extensions—for macOS, Windows, iPhone/iPad, and Android (some support other platforms too) that are under active development and fully supported

Note: My list includes one exception to this rule: iCloud Keychain is available for all the above platforms except Android, but considering how important it is to the Apple ecosystem, I felt it was important to mention it here.

- The capability to sync password data among devices
- A way to autofill passwords into web forms in common browsers on each platform
- A substantial number of positive mentions in professional reviews
- A large number of high ratings from active users
- Except as noted, support for systemwide autofill, Touch ID, and Face ID on iPhone/iPad; systemwide autofill and fingerprint recognition on Android

For each app, I include a brief description and a bulleted list that covers the main features I mentioned in Features to Look For. I include more details and longer descriptions in a few cases where I feel they're especially warranted.

Missing Managers

There being so many excellent password managers, I've dropped a few that appeared in earlier versions of this book (Blur, which has morphed into a different product called IronVest; DataVault Password Manager; eWallet; Master Password [which later became Spectre]; Myki, which has turned into the enterprise-only JumpCloud; and SplashID Pro) because they lack one or more of the attributes I consider essential, and I'd rather not waste your time (or mine) on these when there are better candidates. (I've added a few newer apps to take their place.) I also dropped Password Boss for reasons I explain later; see the sidebar Whither Password Boss?.

In addition, I omit KeePassX. This free, open-source password manager has a lot of fans. But it's for Mac, Windows, and Linux only—there's no mobile version. It offers no sync feature and is quite weak in the area of autofill. There is a KeePass-based password manager called Strongbox that includes autofill and offers both a mobile version and syncing, but it's only for macOS, iPhone, and iPad.

Finally, although I *mention* LastPass, I no longer go into detail about its capabilities, because (as you'll read in the sidebar The Decline and Fall of LastPass), a series of serious security breaches has erased my trust in LastPass and I no longer recommend it to anyone.

1Password

1Password not only comes first in alphanumeric order, it's also my favorite and the one I've relied upon for years. I've met the developers, beta-tested new versions, and worked more extensively with it than with any of the others I describe here. In fact, I like it so much I wrote an entire book about it: *Take Control of 1Password*.

Originally, 1Password existed as a standalone app with optional browser extensions. Nowadays, 1Password's browser extensions contain most of the product's functionality (and can communicate directly with 1Password's cloud service to save and retrieve your credentials), so they can be used without a separate app. However, the 1Password app does add useful features and editing capabilities, so I recommend installing both the app and the extensions on supported platforms.

Tip: For fascinating technical details about the extra effort 1Password goes through to protect your data, read the AgileBits Blog. You might want to start with the article You have secrets; we don't. Why our data format is public and move on to 1Password hashcat strong master passwords.

1Password's password generator (**Figure 2**) lets you choose whether to include numbers and/or symbols, and the character length (up to 100). You can also opt for Memorable, or Diceware-style, passwords consisting of a sequence of lowercase English words (with user-defined separators). 1Password uses much longer word lists than Diceware, making it more secure.

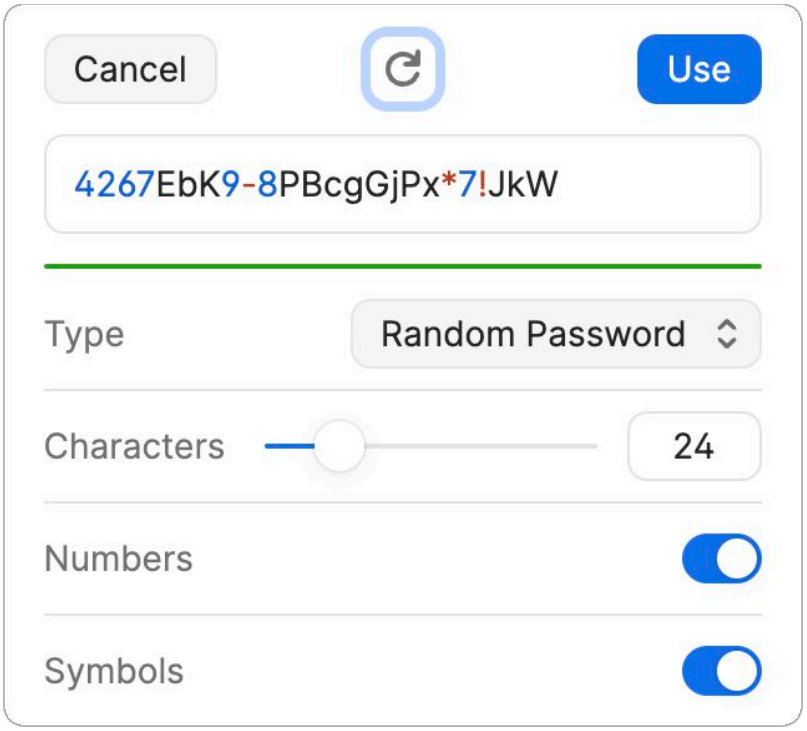


Figure 2: 1Password's password generator gives you numerous options for creating random passwords.

1Password can function as a one-time password generator, replacing standalone apps such as Google Authenticator and Authy (see One-Time Passwords). This eliminates the need to switch to a separate app for one-time passwords when using two-factor authentication or two-step verification—in fact, on most sites, 1Password autofills your TOTP too.

1Password also supports storing, entering, and syncing passkeys, and developer AgileBits is beta-testing a feature that will let you use a passkey in lieu of a master password to unlock your 1Password data.

1Password lets you securely share any item with one or more other people, for periods of time ranging from one hour to 30 days—and those people need not even be 1Password users. If you want a more permanent means of sharing credentials with coworkers or family members, you can sign up for a 1Password Teams or 1Password Families account. These services include cloud-based syncing for team or family vaults, control over who can do what with each shared item, and the capability to add, remove, or suspend team members at any time. If you sign up for one of these services, your subscription fee includes access to 1Password for all team or family members on all platforms, so you won't have to buy licenses separately.

1Password details:

- **Platforms:** Chrome OS, Linux, Mac, Windows, iPhone/iPad, Android, Apple Watch.
- **Extensions (Mac):** Brave, Chrome, Edge, Firefox, Safari.
- **Extensions (Windows):** Brave, Chrome, Edge, Firefox.
- **Autofill/autosubmit:** Autofill with optional autosubmit on keyboard shortcut or click/tap.
- **Form capture:** Automatic (with prompt) as you submit forms.
- **Password generator:** User-defined password length up to 100 characters, with optional digits and punctuation; "Memorable" (Diceware-like) option.
- **Passkey support:** Yes.
- **Data storage:** 1Password accounts (which come in personal, team, and family varieties) use cloud storage but keep a local copy of all data.
- **Syncing:** 1Password accounts use proprietary cloud syncing.

- **Sharing:** Yes.
- **One-time password viewer:** Yes.
- **2FA support:** Yes.
- **Security audit features:** Passive and active. The Watchtower security audit identifies compromised websites (sites for which you haven't changed your password since a known data breach), vulnerable passwords (passwords found in leaked password databases), logins for which passkeys are available but not yet set up, items that may be stored in the wrong account (e.g., personal instead of work), reused passwords, weak passwords, unsecured websites (those for which your login item records a URL starting with "http," not "https"), inactive 2FA (sites that offer two-factor authentication but for which you haven't enabled it yet), expiring items, duplicated items, and SSH keys on disk that haven't been imported into 1Password.
- **Pricing:** The subscription fee for a 1Password account starts at $2.99 per month, billed annually, and includes licenses for all platforms. Subscriptions also include interesting extra features (such as Travel Mode, which removes sensitive passwords from your devices if you're concerned about border inspections) and extensive control over sharing passwords securely.

Apple's Password Tools

Previous editions of this book had a heading called iCloud Keychain that described a set of features built into macOS, iOS, iPadOS, and Safari for managing and syncing passwords. That mechanism still exists, but it now goes by several names, depending on where and how you're using it:

- **iCloud Keychain:** Apple's operating systems have long had a component, called Keychain, for securely storing passwords, passkeys, and other private information. Starting in 2013 with the release of OS X 10.9 Mavericks and iOS 7, Apple made it possible to sync this information across devices using their free iCloud service;

they called this feature *iCloud Keychain*. It makes use of Safari's password generator as well as various tools built into each Apple operating system. Besides handling passwords and passkeys, it lets you store, sync, and fill credit card details; and lets you choose among multiple sets of credentials per website. And, because it integrates with macOS, iOS, and iPadOS, it can also handle your system-level passwords. For example, if you entered a Wi-Fi password on your Mac and later connect to the same network on your iPhone, you won't have to type in that password.

In recent years, Apple has significantly expanded their operating systems' password management capabilities. For example, they now have a TOTP viewer; support importing passwords from and exporting passwords to other password managers; let you store, sync, and enter passkeys (see Authenticate Without Passwords); allow secure sharing of credentials; and, starting in macOS 14 Sonoma, iOS 17, and iPadOS 17 and later, offer extensions that support Chrome, Edge, and most other Chromium-based browsers in macOS.

- **iCloud Passwords:** Another move was to extend support for password syncing and filling to Windows, by way of *iCloud Passwords*, which is part of iCloud for Windows. (After installing iCloud for Windows, you must separately install the iCloud Passwords extension for Chrome or Edge.) There's also an iCloud Passwords Firefox extension, though at present it works only in macOS.
- **Passwords:** In 2024, with the release of macOS 15 Sequoia, iOS 18, and iPadOS 18, Apple made another small but meaningful step, moving the interface for viewing and editing passwords in your iCloud Keychain from System Settings > Passwords (and Safari > Settings > Passwords) to a standalone app called *Passwords*. The underlying capabilities are almost entirely unchanged from previous implementations, but Passwords *looks* more like third-party password manager apps—and because of that, a lot of articles made it sound as though Passwords was, in fact, an entirely new password manager.

By whatever name—iCloud Keychain, iCloud Passwords, or Passwords—Apple's software for managing secure credentials has essentially the same set of features, and it's now available to most people using most browsers on most platforms. And, for people who use only Apple devices (and possibly Windows 11), it could be all the password manager you need. However, it's limited compared to some of the other password managers I discuss here. For example:

- It can't sync with Android.
- On a Mac, some third-party browsers still aren't fully supported.
- Although it does create random passwords, by default they're all 20 characters long with the format `XXXXXX-XXXXXX-XXXXXX`, where each `X` is an alphanumeric character. Because the hyphens never vary (no other symbols are used and the grouping is always the same), the effective length is only 18 characters. (If you look carefully, you can also choose 15-character passwords without the hyphens.) If you want a longer or more varied password, you can edit Apple's suggestion manually, but then you lose the randomness.
- It won't help you with software licenses, secure notes, and other arbitrary data types. (Using the Passwords app on Apple devices, you can, however, add a note to any password.)

For these and other reasons, I use Apple's password tools only as a supplement to a third-party password manager. (You can use both at the same time.) To learn more, read my book *Take Control of iCloud*.

Details for Apple's password tools:

- **Platforms:** Mac, iPhone/iPad, Windows 11.
- **Extensions (Mac):** None needed for Safari; extensions available for Chrome, Edge, and most Chromium-based browsers, plus Firefox.
- **Extensions (Windows):** Chrome, Edge.
- **Autofill/autosubmit:** Autofill with autosubmit on demand.
- **Form capture:** Automatic (with prompt) as you submit forms.

- **Password generator:** By default, all passwords are 20 characters long and follow the same pattern. However, there's an option for 15-character alphanumeric passwords without punctuation characters, and you can manually edit the suggested password if you like.
- **Passkey support:** Yes.
- **Data storage:** Cloud-based (but not accessible via a web browser), with complete local copies on each device.
- **Syncing:** Proprietary cloud syncing.
- **Sharing:** Yes.
- **One-time password viewer:** Yes.
- **2FA support:** Optional 2FA (see Use Apple's Two-Factor Authentication) or 2SV (see Use Apple's Two-Step Verification).
- **Security audit features:** Identifies reused, compromised, and easily guessed passwords.
- **Pricing:** Included with macOS, iOS, and iPadOS; requires free iCloud membership.

Bitwarden

Bitwarden is open-source software, which means anyone can examine the code. In theory, that means bugs and security vulnerabilities shouldn't be able to hide; in practice, there are no guarantees. In my testing, it performed adequately, and though I found it unexceptional in terms of features and user interface, it's an excellent choice for those on a budget or who strongly favor open-source apps. You can also host your own Bitwarden server—a plus for anyone paranoid about cloud storage. Bitwarden supports storing, entering, and syncing passkeys; using a passkey to authenticate the web app; and using a second, trusted device to authenticate the Bitwarden desktop or mobile app.

Bitwarden details:

- **Platforms:** Mac, Windows, Linux, iPhone/iPad, Android.

- **Extensions (Mac):** Brave, Chrome, Edge, Firefox, Opera, Safari, Tor Browser, Vivaldi.
- **Extensions (Windows):** Brave, Chrome, Edge, Firefox, Opera, Tor Browser, Vivaldi.
- **Autofill/autosubmit:** Autofill (without autosubmit) when you press a keyboard shortcut or use an inline menu, and optionally on page load.
- **Form capture:** Automatic (with prompt) as you submit forms.
- **Password generator:** Passwords can be up to 128 characters long and optionally include uppercase letters, lowercase letters, digits, and symbols; you can specify a minimum number of digits or special characters, and omit ambiguous characters (`1`/`l`/`I` and `O`/`0`).
- **Passkey support:** Yes.
- **Data storage:** Cloud or private server, with a local copy.
- **Syncing:** Via a proprietary cloud service *or* your own self-hosted Bitwarden server.
- **Sharing:** Yes.
- **One-time password viewer:** Yes, with the (paid) Premium version.
- **2FA support:** Yes; physical keys are supported only with the (paid) Premium versions.
- **Security audit features:** Passive only, with the (paid) Premium version.
- **Pricing:** Free; premium version ($10 per year) adds several features, including encrypted online file storage. Six-person family plans are also available for $40 per year.

Dashlane

Dashlane is both attractive and capable—it can automatically generate, store, and fill in passwords; it can also store and enter personal data

(such as address and phone number), credit card numbers, and secure notes. It offers import/export from/to various other password managers, including Apple's Keychain. It also now stores, enters, and syncs passkeys. And, like 1Password, it can generate one-time passwords for accounts that use them without requiring a separate authenticator app. Dashlane even has a couple of neat features that the current version of 1Password doesn't, such as a one-step Password Changer, with support for over 500 sites, that updates a password both on the site and within Dashlane (requires a paid subscription), and an Emergency feature that lets you grant other people secure access to your Dashlane data in situations where you're unable to access it yourself.

Like 1Password, Dashlane is packaged as a "web-first" app, in which all you need is an extension for your favorite browser and an account. As far as I can tell, the standalone desktop app Dashlane once offered is no longer available.

All in all, Dashlane is a solid choice, and the cost is comparable to competing options.

Dashlane details:

- **Platforms:** Chrome OS, Linux, Mac, Windows, iPhone/iPad, Apple Watch, Android.
- **Extensions (Mac):** Firefox, Chrome, Edge, other Chromium-based browsers, Safari.
- **Extensions (Windows):** Firefox, Chrome, Edge, other Chromium-based browsers.
- **Autofill/autosubmit:** Autofill on page load with optional auto-submit, enabled by default (but bear in mind the security risks; see Four Autofill/Autosubmit Models).
- **Form capture:** Automatic (with prompt) as you submit forms.
- **Password generator:** User-defined password length and optional inclusion of letters, digits, and punctuation; optional pronounceable setting.

- **Passkey support:** Yes.
- **Data storage:** Cloud-based.
- **Syncing:** Proprietary cloud syncing.
- **Sharing:** Yes; includes an emergency access feature.
- **One-time password viewer:** Yes.
- **2FA support:** Yes, using a one-time password; also supports a U2F device such as YubiKey for unlocking the app.
- **Security audit features:** Passive and active (but more limited in scope than 1Password).
- **Pricing:** Dashlane has discontinued its free version. Dashlane Premium costs $4.99 per month, billed annually; their Friends & Family plan, which supports up to 10 users, costs $7.49 per month, billed annually. (Prices may vary by country; for example, the prices for someone in Canada, *in U.S. dollars*, are $3.33 and $4.99, respectively.)

What About Dropbox Passwords?

Dropbox has its own password manager, Dropbox Passwords, but my feelings about it are decidedly *meh*. It doesn't meet all the criteria I set out at the beginning of this chapter, it's very light on features, and in particular I found its form capture capability to be extremely flaky and inconsistent. My advice: skip it.

Enpass

The first time I saw Enpass, it struck me as a cheap knock-off of 1Password—almost exactly the same look and feel, but fewer features and less polish. Although the app has changed a great deal since then, and has diverged in design from 1Password, it's still missing a lot of the bells and whistles of the app that provided its inspiration. For example, it *does* have a TOTP viewer, but its setup process is awkward; it lacks secure sharing; and although it can autofill logins and credit cards, it

can't autofill contact info. On the other hand, it is super inexpensive! So if you're on a tight budget, it may be worth a look.

Enpass details:

- **Platforms:** Mac, Windows, Linux, iPhone/iPad, Android, Apple Watch, Android Wear.
- **Extensions (Mac):** Brave, Chrome, Edge, Firefox, Opera, Safari, Vivaldi.
- **Extensions (Windows):** Brave, Chrome, Edge, Firefox, Opera, Vivaldi.
- **Autofill/autosubmit:** Autofill on demand with optional autosubmit (enabled by default, which could pose a security risk).
- **Form capture:** Automatic (with prompt) as you submit forms.
- **Password generator:** Passwords can be up to 100 characters long, with uppercase letters, digits, and punctuation optional; pronounceable option; option to exclude ambiguous characters.
- **Passkey support:** Yes.
- **Data storage:** Local storage with optional sync via cloud services such as iCloud, Dropbox, OneDrive, or Google Drive—or folders shared over a network by other means.
- **Syncing:** Encrypted syncing to/from the storage location of your choice; doesn't offer its own cloud storage.
- **Sharing:** Limited and awkward.
- **One-time password viewer:** Yes.
- **2FA support:** Not applicable.
- **Security audit features:** Passive only.
- **Pricing:** Freemium. Free version is fully featured on desktop computers but limited to 25 items on mobile devices. The paid version, which syncs unlimited items across unlimited devices and adds alerts of security breaches, is available either by subscription

($23.99 per year for individuals; $47.99 per year for families of up to six, after a discounted first year) or as a *one-time* purchase of $99.99.

Keeper

In the first edition of this book, I complained that Keeper was far less capable than every other password manager in this list. It lacked browser extensions, for example, and was entirely too cumbersome to use. I'm pleased to say that the app has made tremendous progress since then—the developer has addressed nearly every complaint I made, and then some.

For example, Keeper now has a nicely configurable password generator, support for filling in contact information and credit cards, a history of previous passwords for each login, the capability to fill passwords in desktop apps (not just browser windows), and the option to grant emergency access to up to five people. The desktop app is no longer the hot mess it was a few years ago, and Keeper has both a TOTP viewer and limited security auditing.

In my opinion, Keeper is harder to use than competing products. So, although it's certainly getting better all the time, I'd still say there are better options in a similar or lower price range.

Keeper details:

- **Platforms:** Mac, Windows, Linux, iPhone/iPad, Apple Watch, Android, Windows Phone.
- **Extensions (Mac):** Chrome, Edge, Firefox, Opera, Safari.
- **Extensions (Windows):** Chrome, Edge, Firefox, Internet Explorer, Opera.
- **Autofill/autosubmit:** Autofill on page load.
- **Form capture:** Automatic (with prompt) as you submit forms.
- **Password generator:** Adjustable length (up to 51 characters) and optional uppercase letters, digits, and punctuation.

- **Passkey support:** Yes.
- **Data storage:** Cloud-based.
- **Syncing:** Proprietary cloud syncing.
- **Sharing:** Yes, including an emergency access feature.
- **One-time password viewer:** No.
- **2FA support:** Yes; several methods available, including SMS, authenticator apps, and U2F devices such as YubiKey.
- **Security audit features:** Identifies compromised credentials (paid version only).
- **Pricing:** Keeper costs $34.99 per year for one user, or $74.99 per year for a family of up to five people. (The annual fee also provides web access and secure record sharing.)

The Decline and Fall of LastPass

For many years, LastPass was at or near the top of my list (and everyone else's). It was a full-featured, highly respected password manager with lots of fans. For a long time, it was even free, with an optional Premium version that added some useful features.

Users started grumbling when, in 2019, the price for a Premium subscription rose from $24 to $36 per year, and they grumbled more in 2021 when LastPass announced that free users would be restricted to just one device type—for example, only computers or only mobile devices. But it was still, to all appearances, a solid product.

Then, due to an errant software update in early June 2021, the LastPass extension for Chrome was completely nonfunctional for weeks (see this support thread). That was not the first time something like that had happened, and such a severe problem did raise questions about LastPass's commitment to testing and quality assurance.

But in 2022, things turned even worse. There were two major security breaches resulting in exposure of private customer data. (See Adam Engst's TidBITS article LastPass Publishes More Details about Its Data Breaches.) After learning about these issues, security researchers, major publications, and review sites started cautioning people against using LastPass (see, for example, this post by Jeremi Gosney on Mastodon), and I couldn't agree more.

The number-one priority of any company storing passwords and other sensitive personal data has to be ironclad security, and LastPass has, unfortunately, demonstrated that they aren't up to the task. With regret, I can no longer recommend LastPass to anyone.

NordPass

NordPass—a password manager from the same company that makes NordVPN—has the usual array of features (although not a one-time password viewer) and appears to be both well designed and competently engineered, but it also doesn't distinguish itself from the competition in any significant way. It's hard to make a compelling case for it, but equally hard to argue against it if it happens to strike your fancy.

NordPass details:

- **Platforms:** Mac, Windows, Linux, iPhone/iPad, Android.
- **Extensions (Mac):** Brave, Chrome, Edge, Firefox, Opera, Safari, Vivaldi.
- **Extensions (Windows):** Brave, Chrome, Edge, Firefox, Opera, Vivaldi.
- **Autofill/autosubmit:** Autofill and autosubmit on page load (optional but enabled by default).
- **Form capture:** Automatic (with prompt) as you submit forms.
- **Password generator:** Passwords can be up to 60 characters long; can optionally include lower- and uppercase letters, numbers, and symbols; and can exclude ambiguous characters (`1`/`l`/`I` and `O`/`0`).
- **Passkey support:** Yes.
- **Data storage:** Cloud with local sync.
- **Syncing:** Proprietary cloud syncing.
- **Sharing:** Yes.
- **One-time password viewer:** No.
- **2FA support:** Yes, but only using an authenticator app.
- **Security audit features:** Passive and (limited) active.
- **Pricing:** Freemium. Free version supports one device at a time; paid Premium version lets you remain logged in on multiple devices, adds secure sharing and security audit features, and costs $44.85 per year (with discounts for the first year and for multi-year subscriptions).

Whither Password Boss?

Early versions of this book included a discussion of Password Boss here. While working on the third edition of this book, when I downloaded the then-latest version of the app, installed it, launched it, and tried to log in, the app crashed on my M1 Mac before I even had a chance to enter my password. This happened over and over again. That was likely a fleeting bug, but I need my password manager to be bulletproof, so those crashes were enough for me to recommend keeping your distance from this app. Since then, the company has pivoted to a largely enterprise focus, which gives me another reason not to recommend the app to consumers.

Proton Pass

Proton Pass is an up-and-coming, open-source password manager from Swiss company Proton, best known for their encrypted Proton Mail email platform but also responsible for encrypted calendar, cloud storage, and VPN services. Proton Pass is currently available in the form of a web app, browser extensions, a mobile app for iPhone/iPad or Android, and a Windows desktop app, with macOS and Linux desktop apps "coming soon." (The desktop apps are available only with paid plans.)

Based on my experience with the company, I have no reason to worry about issues of security or privacy. However, Proton Pass has fewer features than more mature apps such as 1Password, Dashlane, and Roboform, though it's actively under development and gaining new capabilities regularly. What you get with the free version is more than what some paid password managers offer, and if you already subscribe to the company's Proton Unlimited suite, getting extra password manager features as part of the deal is hard to argue with.

Proton Pass details:

- **Platforms:** Mac, Windows, Linux, iPhone/iPad, Android.
- **Extensions (Mac):** Brave, Chrome, Edge, Firefox, Safari.
- **Extensions (Windows):** Brave, Chrome, Edge, Firefox.

- **Autofill/autosubmit:** Autofill on demand; no autosubmit.
- **Form capture:** Automatic (with prompt) as you submit forms.
- **Password generator:** Passwords can be up to 64 characters long; can optionally include lower- and uppercase letters, numbers, and symbols. Memorable (Diceware-style) password option.
- **Passkey support:** Yes.
- **Data storage:** Cloud with local sync.
- **Syncing:** Proprietary cloud syncing.
- **Sharing:** Yes. Share with up to three people using the free plan, or up to 10 with a paid plan.
- **One-time password viewer:** Yes, for paid plans only.
- **2FA support:** Yes, but only using an authenticator app.
- **Security audit features:** None.
- **Pricing:** Freemium. Free version supports unlimited logins and syncing across unlimited devices, 10 disposable "hide-my-email" addresses, and vault sharing for up to three people. Pass Plus ($4.99 per month if billed monthly or $1.99 per month if billed annually) adds unlimited hide-my-email addresses, vault sharing for up to 10 people, a one-time password viewer, and other features. Also available as part of Proton Unlimited ($12.99 per month if billed monthly or $9.99 per month if billed annually), which adds encrypted email, cloud storage, calendars, and a VPN.

RoboForm

RoboForm is a well-designed, professional-looking tool with a robust feature set (**Figure 3**). It includes extensions for the usual range of browsers; within a browser window, you can enter credentials (including a one-click autofill and autosubmit option), generate passwords, save forms, search and view your logins, and so forth. My biggest complaint is that filling in your credentials on a Mac is unnecessarily awkward. If you use the systemwide menu, which is (after a fashion)

accessible from the keyboard, it always opens a new tab. If you use the browser extension, it always seems to require one or more clicks; there's no way to fill in (but not submit) a form on an open webpage using just the keyboard, to say nothing of autofill on page load. (A RoboForm rep told me that they consider the Windows version to be their flagship product, with a full range of features; versions on other platforms are somewhat less capable.)

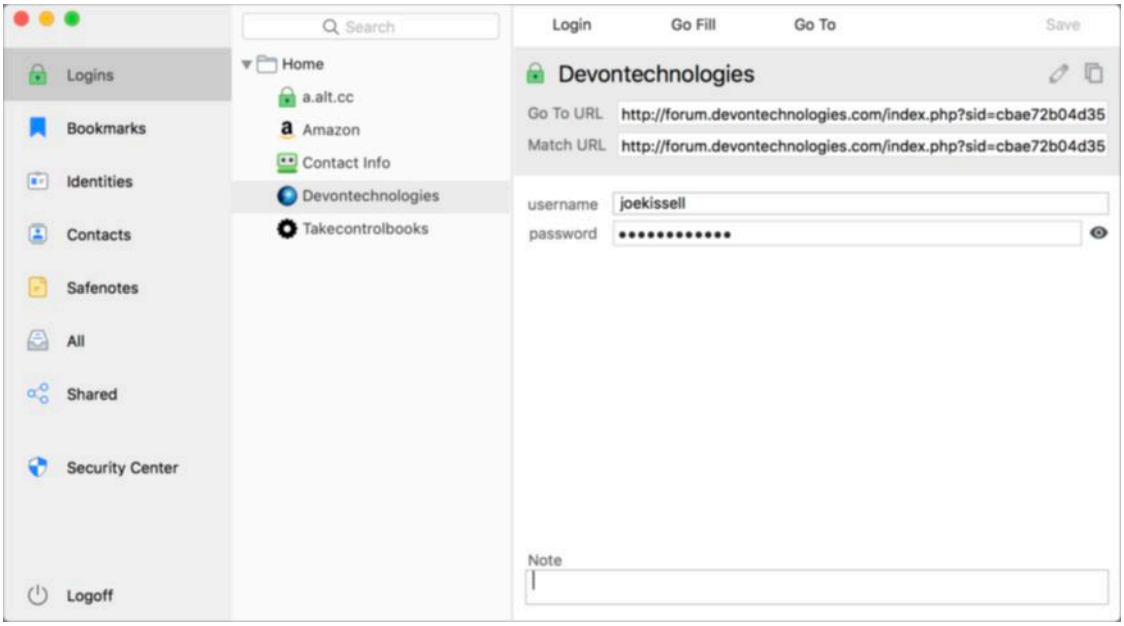


Figure 3: RoboForm's main window has an uncluttered interface for entering and editing login data and other personal information.

At your behest, your credentials can be synced automatically with RoboForm's proprietary (and paid) cloud service, RoboForm Everywhere. That's the only sync option available, but it worked well in my testing. The password generator lets you do the normal things—specify password length, upper- and lowercase letters, digits (including a minimum number of digits), and your choice of punctuation. You can also exclude similar-looking characters (`1`/`l`/`I` and `O`/`0`) and, in a feature that I've not seen elsewhere, you can opt for hexadecimal passwords (that is, only the digits 0–9 and the letters A–F).

RoboForm isn't flashy, but it covers all the basics and then some—including a basic (passive) security audit. Like Dashlane and Keeper, it has an emergency access feature that enables you to give one or more

trusted contacts access to your passwords after a designated waiting period.

Although RoboForm is a solid performer in most senses, I have difficulty recommending it for Mac users because the Mac version lacks important features available in the Windows version, not the least of which is the crucial usability element of letting you fill in your credentials for the page you're currently viewing without having to click that site's name in the RoboForm menu or toolbar.

RoboForm details:

- **Platforms:** Mac, Windows, iPhone/iPad, Android.
- **Extensions (Mac):** Chrome, Firefox, Safari.
- **Extensions (Windows):** Chrome, Edge, Firefox, Opera.
- **Autofill/autosubmit:** Autofill (with or without autosubmit) when you click a button or choose a site's name from a RoboForm menu in your browser. (You can assign a keyboard shortcut to display a menu, but not to fill in your credentials or log you in.)
- **Form capture:** Automatic (with prompt) as you submit forms.
- **Password generator:** User-defined password length and inclusion of upper- and lowercase letters, digits, and punctuation; options to exclude ambiguous characters (`1`/`l`/`I` and `O`/`0`) and restrict characters to hexadecimal (0–9 and A–F).
- **Passkey support:** Yes.
- **Data storage:** Your choice of local (RoboForm Free) or cloud-based (RoboForm Everywhere).
- **Syncing:** Proprietary cloud syncing, with paid version only.
- **Sharing:** Yes, via shared folders, including an emergency access feature.
- **One-time password viewer:** No.
- **2FA support:** Optional support for a one-time password via email.

- **Security audit features:** Passive only.
- **Pricing:** Freemium. Free version doesn't sync across devices. Subscriptions to RoboForm Premium, which add cloud syncing and several other features, include all platforms and cost $29.88 per year. Family subscriptions, which cover five users, cost $47.75 per year.

Sticky Password

Sticky Password is another password manager that looks fantastic on paper, but is only so-so in real life. It omits important features found in competing apps (such as a TOTP viewer and passkey support). On the plus side, the interface has become much more polished in recent years, and the password generator, which was previously unconfigurable, now has a respectable array of options.

Sticky Password details:

- **Platforms:** Mac, Windows, iPhone/iPad, Android, others.
- **Extensions (Mac):** Brave, Chrome, Firefox, Safari.
- **Extensions (Windows):** Brave, Chrome, Edge, Firefox, Internet Explorer, Opera, others.
- **Mobile features:** iPhone/iPad: autofill, Touch ID, and Face ID. Android: fingerprint recognition and integration only with Firefox and Dolphin (credentials available in some other browsers and apps via a floating window).
- **Autofill/autosubmit:** Autofill on page load, with optional autosubmit.
- **Form capture:** Automatic as you submit forms.
- **Password generator:** Passwords can be up to 99 characters long; can optionally include lower- and uppercase letters, numbers, and symbols; and can exclude ambiguous characters (`1`/`l`/`I` and `O`/`0`).
- **Passkey support:** No.

- **Data storage:** Cloud-based.
- **Syncing:** Proprietary cloud syncing.
- **Sharing:** Yes (Premium only).
- **One-time password viewer:** No.
- **2FA support:** Yes, including biometrics on mobile devices and an optional one-time password via email—but only for initial authorization of a new trusted device.
- **Security audit features:** Passive and (limited) active.
- **Pricing:** Freemium. Basic features are free. The Premium version costs $19.99 per year or $59.99 for a permanent license; it offers cloud or local Wi-Fi sync, cloud backup, secure sharing, and priority support—plus a donation to save endangered manatees (seriously).

Joe's Recommendations

Of the third-party password managers I've tried, the ones I consider top-tier contenders are 1Password, Dashlane, and Bitwarden. (I previously included RoboForm on that list, but now feel that its less convenient autofill bumps it down just a notch.) While I'm impressed by the progress Keeper and Sticky Password have made, I don't feel those can quite match my top choices, but that might change in time. Apple's Password Tools are a good choice for what they do—especially for iPhones and iPads—but I think of them as a supplement to, rather than a replacement for, a standalone password manager.

I've mentioned that 1Password is my favorite; the interface and feature set most closely match my needs. However, any of the apps in the previous paragraph should meet most people's needs.

As for the other password managers I covered in this chapter, I think they're less capable and less convenient, but if you happen to use and like one of them—or any of the numerous options I didn't mention—more power to you. I'd much rather you use a tool you feel comfortable with than struggle with something else only because I recommend it.

Keep Your Passwords Secure

If you stored your fortune in a safe deposit box, you wouldn't keep the key hanging on a hook outside your house. The same should be true of your passwords: if you keep them written on a whiteboard by your desk, they're not safe. But even if you don't write them down, there are many ways someone might discover your passwords.

In this chapter, I look at some of the ways your passwords might fall into the wrong hands, and give you tips on keeping them safe. I also discuss backing up your passwords and devising a plan to ensure that your passwords are available in case of emergency.

Avoid the "Weakest Link" Problem

Suppose you have a fantastic password that would take the world's best supercomputers centuries to crack. You've stored the password in your password manager, which uses a weaker master password that's easier to remember. And because you still worry that you might forget it, you store your master password in an unencrypted text file on your hard disk. You can see where I'm going with this: you've nullified the security of that great password, because someone can get to it by way of the text file that unlocks your password manager, without any guessing or cracking. Even without that file, your super-strong password is reduced to the strength of your master password.

Just as a chain is only as strong as its weakest link, a password is only as strong as the weakest means by which someone can (directly or indirectly) get to it. That concept is straightforward enough, but consider some of the ramifications:

- If you write down a password, the password (and whatever it protects) is only as safe as the written copy. As I explain shortly,

that doesn't mean you should *never* write down your passwords, but if you do, you'd better take extraordinary care to protect those written copies.

- If you click a "forgot my password" link and a site emails you your password or a link to reset it, that password is only as safe as the password used to access your email account (and possibly much less secure, since there are ways a determined attacker could access your email account even without its password).
- If you type the password into an encrypted file on your computer (or, better yet, encrypt the entire disk), the password is only as safe as the password protecting the encrypted data—and that depends further on the encryption method used, since some methods are easier to crack than others, regardless of the password strength.

Taking all these situations into account, my advice is:

- If you write down any of your passwords, keep them in a very safe place (such as on your person). For increased security, modify them in some way (such as reversing the order of the characters)—but don't forget how you modified them! For ideas about writing down passwords that someone else may need to access, read Prepare an Emergency Password Plan, later in this chapter.
- When typing your passwords, especially on a phone, make sure no one watches over your shoulder to see your screen or the keys you press.
- Take appropriate precautions when using wireless networks, such as turning on a VPN when using public Wi-Fi.
- Make all your passwords equally strong (that is, make sure they all have high entropy).
- Store your passwords in a password manager (protected with a strong master password, of course). Lock your password manager when not in use, and back up your data (see Back Up Your Passwords, shortly ahead).

Tip: If you use an iPhone or iPad, also make sure prompts for your Apple Account password are really coming from Apple! Read the article iOS Privacy: steal.password - Easily get the user's Apple ID password, just by asking for details.

Back Up Your Passwords

Consider this nightmare scenario: You've carefully created hundreds of nicely random passwords and stored them in your password manager so you don't have to remember them. But then your computer crashes and your data file is damaged. Or your computer is stolen. Or any of a dozen other catastrophes occurs, and your stored passwords are lost. Unless you have a photographic memory, the best defense against all these situations is a good backup.

Note: Good backups are essential if your password manager stores its data only on your devices. If it uses a cloud service, presumably the provider of that service backs up the data on their servers regularly, which should vastly reduce your risk of data loss. Even so, if you have the option to back up your passwords separately, do so!

Ideally, your passwords will be backed up automatically as part of your regular backup regimen and no additional steps will be necessary. Just make sure whatever backup method you use includes your password manager's data file. Since your password manager already encrypts your passwords, you need not worry about going to the extra bother of encrypting your backups too. (There are other good reasons to encrypt your backups, but don't do it just for your passwords.)

Of the many ways in which one can back up a computer, I'm especially fond of cloud backups because they require no hardware, can be set to run continuously without any intervention, and can be restored even if every device in your home or office is stolen or destroyed. I use Backblaze, but IDrive and dozens of other services offer similar capabilities.

You might also store just your encrypted password manager data file in the cloud using Dropbox, iCloud Drive, or another service that lets you

access stored data as though it were a local folder and offers versioned backups.

I won't try to talk you into a particular method of backing up your passwords, but however you choose to do it, *back them up!*

Tip: You can read a detailed review of top cloud backup providers in the Wirecutter article The Best Online Cloud Backup Service. If you're a Mac user, check out my book *Take Control of Backing Up Your Mac* for comprehensive backup advice.

Prepare an Emergency Password Plan

Suppose you've stored all your passwords securely in your password manager—but you've kept your master password *only* in your head. Then one day you're in a terrible accident. While you lie unconscious in a hospital bed, your spouse, employer, or attorney urgently needs access to something protected by one of your great passwords—a bank account, insurance records, your email, or whatnot. We don't like to think about such eventualities, but they do occur. If you become incapacitated or die, how will someone else be able to reach your password-protected data?

This isn't merely a hypothetical situation. In February 2019, the news broke that a Canadian cryptocurrency firm called QuadrigaCX had lost CDN$190 million after the death of its founder—who kept the password to the company's offline storage *only in his head*. As far as anyone knows, he never had an emergency plan that would enable anyone else in his company to learn that password. Without a way to recover the encrypted data, the company was forced to declare bankruptcy, and a great many customers lost their money.

Several password managers—including Dashlane, Keeper, and RoboForm—have a built-in emergency access feature to avoid situations like this. Although the details vary from one app to the next, the basic idea is the same. You designate one or more emergency contacts ahead of time, each of whom will typically need to set up their own copy of the

app in question and their own (usually free) account. You also set a time delay, such as five days. If something happens to you, your emergency contact can request secure access to your passwords. The provider attempts to notify you, but unless you take action to prevent the other person from accessing your passwords within the time limit you set, they will, at that time, be able to unlock your password manager—without needing to know your master password.

Tip: I go into great detail about how to ensure that the right people have access to the data that you want to preserve after your death in my book *Take Control of Your Digital Legacy*.

Alternatively, if you use a password manger that supports secure sharing—for example, 1Password Teams or 1Password Families—you can use it to give your spouse, a coworker, or another trusted person access to your key passwords. The main downside is you can't restrict this access to circumstances when you're unresponsive. You'll also have to choose which passwords you want someone else to access and which, if any, you'll keep completely private. (AgileBits is reportedly considering adding an emergency access feature to 1Password.)

Otherwise, your best protection may be a simple piece of paper.

I don't recommend writing down all your passwords, but I do suggest writing down at least the master password for your password manager or, perhaps, all the passwords on your short VIP list—and including instructions for using them. (For example, write down instructions for logging in to your computer and opening your password manager, or locating a secure copy of your password list in the cloud.)

The instructions themselves need not be kept especially secure, but the password(s) must be—try one of the following:

- Keep the password list in a safe deposit box, and make sure a trusted loved one has the key.
- Ask your loved one to memorize your master password. (A periodic practice run to find and unlock your password manager is smart.)

- Write down your important passwords (or just your master password) in an obscure location, but one that both you and a loved one can easily remember. You probably don't have to make it super-spy-proof, just hidden from a casual thief who isn't particularly looking for it.

  Here are some examples:

  - Open your favorite book to the page corresponding to the last two digits of the year you were born (or some other memorable date). Lightly, in pencil, write the password vertically on the inside edge of that page, near the spine.
  - Write the password on a recipe card, in the middle of the most complicated recipe you can find. For example, if you have a recipe for mole sauce, it might include "1 tsp ground cinnamon; 1 tbsp s8#gUl4Bx5; 3 tbsp ground sesame seeds…"
  - Put your passwords in a notebook that you keep in a box in your attic or basement, such as the one holding holiday decorations.
  - Use a simple cipher or code to write down your master password. For example, substitute the next higher letter or number for each one in your password (`gU#7f.9t3vQd` becomes `hV#8g.0u4wRe`). Although this won't stop a cryptanalyst, it will confuse or at least slow down most thieves.

I want to reiterate that, in terms of security, writing down passwords always exposes you to a certain amount of risk. Realistically, however, in most cases this risk is fairly small; except in highly unusual circumstances, thieves will be far more interested in stealing your computer than in turning your office upside down to find a password. In general, the greater risk is having your passwords be inaccessible when you need them.

Regardless of which tactic you choose (and no doubt you can be even more creative!), the key is to discuss your plan thoroughly with the person who would need access to your passwords in case of emergency, just as you would an insurance policy or a will.

Avoid Password Phishing Scams

If you're like most people, you frequently receive email or SMS messages claiming to be from a bank or from sites that process financial data such as PayPal, eBay, or Amazon.com. The messages often urge you to "update" or "verify" your account, warning you that if you don't, you'll suffer dire consequences (such as having the account disabled). Sometimes they state that some item you didn't order is about to be shipped and charged to your account, or that a payment or money transfer didn't go through. Invariably the messages ask you to click a link; if you do, you're taken to a website that asks for your username (or account number) and password.

The websites, and the messages leading to them, look authentic. They use the same fonts, logos, and layout that you'd expect from the company in question. The messages usually have a From address at the real company, too. But, in fact, the whole operation is a scam designed to trick you into giving away your username and password to criminals. (They're fishing for data—hence the nickname *phishing*.) As soon as you enter your information, the people running the site will try to log in to the real bank or website with your credentials. With access to your account, imagine the damage they can do: they can steal not only your money but also your identity.

You can often tell if a message is a phishing attempt by looking at its source, or raw, unformatted contents (your email client probably has a command to display this). In the source, locate the URL you've been asked to click (which may be different from the URL in the visible link). You'll probably see that it's a numeric IP address (such as 123.45.67.89) or a misleading domain name (such as http://www.paypal-upgrade.net). In some email clients, there's an even easier way—you can hover your pointer over a link, without clicking, to see its destination.

When in doubt, don't click a URL in such a message. If you think a company might legitimately want you to update your account—unlikely as that is—go to its website by typing the address into your browser. If, after logging in, you see no notices about needing to take any action, you can assume the message was a phishing attempt.

Audit Your Passwords

Perhaps, upon reading this book, you realize that some or all of your passwords are terrible, and you're committed to choosing and using good passwords from now on. Fantastic—but what about all those existing passwords, which may number in the hundreds? How do you find the bad ones and change them? You need to audit your passwords to determine how bad the problem is and where fixes are needed.

Several password managers, including 1Password, Dashlane, and RoboForm, have a security audit feature (which goes by various names) that identifies weak, reused, or otherwise vulnerable passwords in your vault. In some cases, the feature calls special attention to passwords for sites with known security breaches that haven't been changed since the breach became public, or credentials that show up in leaked lists that have been made public, among other dangers. These tools can help you find and address the most serious problems first.

Dashlane goes a step further to help deal with passwords their security audit has identified as being problematic. It has a feature that lets you change passwords—even a long list of passwords—in a single step. That is, instead of logging in to each site, going to the account page, clicking the Change Password button, generating and entering a new password, and recording it in your password manger, you just click a button that says "Do all that stuff for me, for all the selected passwords," and it happens automagically. There is a catch, however, in that this feature works only with a limited number of sites—those for which Dashlane already knows the specific password-change steps. That might be just a small fraction of the passwords you need to change.

For password managers without a feature like this—or for sites the automatic change feature doesn't support—there's no quick or easy way to change many different passwords at once, so brace yourself: this is going to be a bit of a slog. But you can make the process manageable by following the steps in this chapter.

Understand the Overall Process

Changing a single password might take you a minute or two, but changing your password for every website where you entered `abc123` over the past 10 years is a pretty big undertaking. If you have tons of so-so passwords, you might feel like you should take a week off work, prepare a few gallons of strong coffee, and plow through the enormous process of changing them all at once. And then, realizing how implausible that is, give up and do nothing at all!

I'm a "something is better than nothing" kind of guy, and I'd rather you take small steps toward having somewhat better security now than do nothing in the hope of eventually getting around to having fantastic security. So, first of all, take a few deep breaths. This big task can be broken down into manageable steps, and those steps can be prioritized so that you deal with the most serious problems first, until eventually you have the whole mess cleaned up. Here's what I suggest:

- First, Look for Weak Passwords in order to determine which ones are most likely to be easily guessable or hackable. Make a note of those—but if you've been using poor password practices for a long time, your note might just say "all of them"!
- Working from the list of passwords you consider too weak, Triage Your Passwords—determine which ones pose the most serious security risk right now, which ones are important but not urgent, and which pose a small enough risk that you can put them off until later.
- Next, starting with the most critical one, Update a Password. While you're at it, you'll want to Check Your Security Questions and Answers and Check the Password Reset Procedure. You'll also have to Update Apps and Devices where the original password might have been stored.
- Repeat this process as needed—perhaps changing three or four passwords a day until your entire collection of passwords has been updated.

Look for Weak Passwords

Your first auditing step is to figure out which of your passwords are too weak to provide reasonable security. If you've read What Makes a Good Password? you should have a good idea of what you're looking for—anything that doesn't meet those criteria! In particular, you want to find any passwords that are:

- **Too short:** Anything under 10 characters would seriously concern me, although I generally recommend 12 characters as a minimum safe length for *random* passwords—and longer is even better.
- **Too simple:** Words you find in a dictionary, keyboard patterns, simple modifications (replacing letters with numbers), and so on make a password easy for a computer to guess, even if it's not short.
- **Reused:** If there are passwords you've used in more than one place, you should change them so that every one is unique.

> **Note:** You may also have passwords that are both strong and unique, but that have been compromised in a data breach; I'll come back to that topic later, in Check for Compromised and Vulnerable Passwords.

The process of finding passwords like these depends on your current approach to storing them. If they're on a piece of paper or in a text file, for example, a simple glance should tell you. If they're in a password manager that displays a strength meter next to each password or offers a security audit feature (as described earlier), that provides a good starting point. The worst case is that you'll have to look at each password individually—in a web browser (if you were using it to save passwords), your password manager, or some other location. (Some tools, such as Apple's Keychain Access, make that process infuriatingly awkward—you have to double-click an item, select Show Password, enter your Keychain's password, and click Allow, potentially hundreds of times. Ouch!)

Tip: If you're using an Apple device, there's an easy way to find reused passwords. On a Mac running Mojave or later, open Safari, go to Safari > Settings/Preferences > Passwords, and authenticate. An alert ⚠️ symbol indicates reused passwords. On an iPhone or iPad, go to Settings > Passwords and authenticate when prompted. Tap Security Recommendations to see reused passwords (as well as those that are easily guessable).

However you do it, your goal should be to locate and mark or make a note of those that are too weak—or even those that are strong enough but that you'd like to make stronger just for good measure.

Although I could have told you to change them as you go, I think that in most cases this identification process will be quick, whereas changing a large number of weak passwords can take a while. That's why, instead, I recommend an intermediate triage step, to which I now turn.

Triage Your Passwords

Start by dividing your too-weak passwords into several groups. Your goal is to deal with the riskiest passwords first and work your way down. I suggest this order, although you may have more or fewer groups, and you may assess your risks differently:

1. Any items on your VIP list
2. Your device passwords/passcodes and passwords for your Apple Account(s) and Google account(s), if not on your VIP list
3. The password(s) for whichever email account(s) you might use for password reset messages
4. Passwords for banks and any other companies that handle money or store your credit card details—Amazon.com, eBay, PayPal, utilities, and so on

Note: Most financial institutions limit your potential loss in the event of fraudulent access to your account, as long as you report it promptly.

5. Passwords for cloud backup services, photo storage sites, or any other services that hold especially valuable personal or business data
6. Passwords for sites that store personal information about you—Facebook, airlines, car rental companies, etc.
7. Everything else

Most of your critical passwords should be covered with the first six items above. Once you have a prioritized list, work through it one password at a time—tackling a few per day—following the process I describe next.

Update a Password

Every site, service, and app has its own procedure for updating passwords. In general, you'll log in with your existing credentials, navigate to a profile or settings page, enter and repeat your new password, and click Save—but additional steps may be required, such as answering security questions and clicking links in verification email messages.

When you get to the point of entering a new password, fire up the password generator in your password manager. Make sure it's configured to generate the longest and most complex password the site or service will accept—assuming you can tell what the rules are. (If you can't, start with a 16-character random password including all possible character types, and scale it back if you get an error message.)

Most password managers can automatically fill newly generated passwords into web forms and save them in your password database for you. If yours doesn't—or if you're not sure—take any necessary steps to confirm that your new password is entered in the web form as well as safe and sound in your password manager.

Reminder: Dashlane has a great feature that logs you in to a site with your old password, goes to the page for changing your password, generates and enters a new password, and updates the password manager's database—all in a single step. I hope to see a similar feature in a future version of 1Password.

Now your password is changed locally and online, but before you move on to the next password, I recommend three other steps.

Tip: Before changing your passwords—especially your computer's login password or your password manager's master password—be sure to back up your data. If something should go wrong when changing the password, you can restore your old data and try again.

Check Your Security Questions and Answers

Not all sites and services use security questions, but the number is growing. While you're in the settings area, check if you previously answered any security questions. If you did—and if you told the truth—take another moment to change the answers to lies (as I covered in Handle Security Questions) and to record both the questions and your fake answers in your password manager. Doing so adds considerable security, because it will prevent people who can guess or Google common facts about you from resetting your password.

Check the Password Reset Procedure

This step may be the trickiest. Your mission is to find out what this site or service does when people lose or forget their passwords—but without following the procedure, because then you might be forced to change your password again. If it's not stated in an obvious place, click around in the online help pages or the FAQ, and you'll probably find a description of the password reset procedure.

Why go to this bother? First, you want to be sure that if the site uses a separate email address to send your password reset link or instructions, the address the site has on file for that purpose is the right one—

by which I mean, ideally, the special, secret address you set up earlier (see Manage Email Options) just for password resets. If you had another address entered, this is the time to change it.

Second, if you encounter one of those rare sites that uses another technique—such as calling you on the phone or sending an SMS message—you can make sure it has your current contact information. I've occasionally gone through this process with a site I haven't visited in a while only to find that the phone number on file is one I haven't used in years, so password resets would have failed.

Tip: Yet another thing you might check while you're here is your mailing address, for those sites that store it, if you've moved since you set up the account.

Update Apps and Devices

Sometimes, updating your password in one place has a cascading effect. Let's say you've just changed your Netflix password. Netflix knows the new password, and so does your password manager. But what about your Apple TV device, Roku, PlayStation, Xbox, Blu-ray player, DVR, smart TV, or any other device that might connect to your Netflix account? What about the Netflix app on each of your mobile devices? You must now go to all those other locations and update your password there, too.

I chose Netflix as an example because it's one of those services you might access from lots of different places. Gmail, iCloud, Dropbox, and other cloud services fall into the same category. But you probably won't have too many passwords that require you to update so many different apps and devices. If you're anything like me, 90% or more of your passwords are for websites that don't connect to anything else. But you should take a moment to make sure your new password is entered everywhere it needs to go.

As you work through these changes, remember to Avoid the "Weakest Link" Problem. Sometimes an app or device offers to remember a password for you—but is not, itself, protected by a strong password.

It's not a big deal to save your Netflix password on your PlayStation, because the amount of account-related damage that could occur if someone stole the PlayStation is small. But saving your email account password on an iPhone without a passcode enabled is a far bigger security risk. Always think through the implications of what someone could learn or do if they got hold of your device—and then take whatever steps are appropriate to protect yourself.

Should You Change Your Passwords Regularly?

Some sites and services force you to change your password periodically. I can *almost* see the wisdom in such a policy. One positive effect is to reduce the chances that you'll reuse another password; another is to minimize the risks for people who routinely use awful passwords. Fair enough—but because you've read this book, you're already too smart to choose poor passwords or reuse a password.

From what I've seen, when passwords are leaked or stolen, thieves tend to try them immediately (in case someone discovers the security breach and alerts users or forces a systemwide reset). So if your password hasn't been compromised already, then changing it today won't increase your security. However, if it was compromised yesterday, then the thief probably already broke into your account, so changing it now (as you naturally would after discovering it had been stolen) is closing the barn door after the horse has bolted. (For more reasons frequent password changes are bad, read Brian Barrett's Wired article Want Safer Passwords? Don't Change Them So Often.)

It's essential to change your password immediately after a known or suspected breach, but the only situation in which I think a periodic password change will help is if a theft happens to occur right before you change your password but the thief doesn't get around to trying your password until just after. The odds of such a situation occurring are slim, so unless a system obligates you to change your password, I don't recommend doing so periodically "just in case."

In the event of a security breach, you should also deauthorize third-party sites and services that were authorized with the compromised account; see Authenticating with Another Site's Credentials.

Check for Compromised and Vulnerable Passwords

If a password is too short, too simple, or a duplicate of another password you use, you know it's bad news no matter what. However, even the world's longest and most random password is worthless if somebody who shouldn't have access to it knows what it is! And that's why, after dealing with the low-hanging fruit as we just did, it's a good idea to take the next step and look for otherwise good passwords that have potentially fallen prey to the bad guys.

Although different password managers use words like "compromised" and "vulnerable" somewhat inconsistently, what I'm referring to here are two different classes of danger:

- **Sites with known data breaches:** When a website—especially one run by a huge company like Adobe or Facebook—has a security fault that may have enabled someone from the outside to access customer data such as login credentials, that's a *data breach*. Breaches can occur for a wide variety of reasons, including hacking, software bugs, theft, and human error. The nature of data breaches is such that the company might be unable to tell exactly what data, if any, was exposed—they might know only that some or all of their data was potentially unsafe for a period of time.

  Because you can't know for certain whether *your* password was compromised in the data breach, the only smart assumption is that it was—meaning you should change that password immediately! And how will you know if a company storing your credentials has had a data breach since the last time you changed your password? Well, companies you do business with *should* inform you of breaches. Beyond that, you can keep an eye on the news, or search in the massive (although incomplete) database of the Privacy Rights Clearinghouse. Better yet, if your password manager has a security audit feature that tracks breaches (such as the Compromised Websites category in 1Password's Watchtower feature), you can simply check there.

- **Credentials appearing in a leaked list:** A fraction of the time data breaches occur, the responsible parties download as many usernames and passwords as they can—sometimes they're still encrypted, but all too often they aren't. Lists of stolen credentials have sometimes numbered in the *hundreds of millions*. Sometimes the thieves sell the lists, or share them publicly, but one way or another they usually make their way into the hands of security researchers. (That's how companies are able to compile those lists of terrible passwords I mentioned back in Threat #2: Guessing.)

  If any password you currently use is on one of those lists, it's worthless forever, because it's already available to be used in a dictionary attack. You should change it immediately. Sometimes your email address may appear in one of these lists, but alongside a still-encrypted password. That's slightly less worrying, but you shouldn't assume the encryption will never be broken. Change your password anyway—right now!

  To see whether your email address (or username) appears in any public lists of hacked passwords, visit haveibeenpwned.com; you can separately check any of your passwords at the site (it doesn't store usernames and passwords together, for obvious security reasons). That database is also integrated into 1Password, via the optional Check for Vulnerable Passwords category of its Watchtower feature. See 1Password for more details about Watchtower. (Some other password managers also have ways of alerting you to leaked credentials as part of their security audit feature.)

Authenticate Without Passwords

In three previous editions of this book, I criticized the breathless proclamations we've heard for years that promised a passwordless future—apps, devices, services, or other technology that would somehow "kill" the password and put us out of our collective misery by giving us an easier yet more secure way to log in. We've heard this story for decades, but reality has never come close to matching up.

Until recently, most "passwordless" authentication methods were merely convenience features layered on top of passwords. Behind the scenes, you still had to *have* a password for nearly every site and service, even if you never *saw* or *typed* it. (Magic links, described earlier in What About Magic Links?, fall into this category.) And the truly passwordless authentication methods that did exist relied on special hardware, making them useless for most people and situations.

But now, finally, a real solution is at hand. I could not be happier to report that the very smart people at the world's top technology companies have devised a technology called *passkeys* that can be used almost anywhere. It has started to appear on everyday websites that normal humans use, and—assuming you have suitably recent hardware and software—it's a joy to behold.

The first time I tried passkeys, I registered for an account on a new website. I picked a username but was never asked for a password at all. And I don't mean I used a password manager to create and fill in a password—I mean there literally wasn't one. All I did was rest my fingertip on my MacBook Air's Touch ID sensor, and the account was created. I signed out, signed back in, and again, all I had to do was touch a sensor. No typing, no menus, no keyboard shortcuts, and no additional software—just my browser. I've tasted the passwordless future, and it is *amazing*. (The experience is similar on other platforms, but if you're using a device without biometric capabilities, you'll

still have to type your device's login password or passcode/PIN to authenticate when using a passkey.)

Passkeys are an implementation of an industry standard called Web-Authn, and they're not just about convenience. Even more importantly, passkeys provide significantly greater security than passwords, with less effort. (I'll come back to the security benefits in a moment.) Easier *and* safer: what's not to like?

Now, I know this sounds exciting—and it is—but let me temper your enthusiasm up front. WebAuthn in general and passkeys in particular have tremendous potential, and I explain both how this all works and how to use it just ahead. But as of early 2025, support for this technology is still in its early phases. Most people will find they can use passkeys in only a handful of places right now, but over the coming years, I expect rapidly growing adoption.

How Passkeys Work

Take Control author Glenn Fleishman wrote a lengthy article for TidBITS: Why Passkeys Will Be Simpler and More Secure Than Passwords. If you want the complete story, with all the technical details, read Glenn's article; it's still largely accurate even though it was written back in mid-2022. Here, I offer just a quick summary.

The underlying technology, WebAuthn, was first created in 2016, and has been available to the public in various forms for quite some time. But until recently, most people needed a special third-party hardware device (usually a FIDO2-compatible USB key; see Physical Keys) to take advantage of WebAuthn. What Apple, Google, Microsoft, and others are doing now is eliminating the need for a separate hardware key and building support right into operating systems and browsers.

WebAuthn relies on public-key cryptography, in which a user has a pair of cryptographic strings (picture them as long, random passwords) called keys. One is private, meaning it stays on the user's device(s) and is never shared, while the other is public, meaning it can be shared freely with no adverse security implications. (And don't

worry, you don't have to create these keys manually; your device, browser, or password manager does this for you, automatically and instantly, when needed. You never have to see or directly interact with the keys.) This *pair* of keys is what we refer to as a passkey.

The interesting twist is that if someone encrypts data with the freely available *public* key, it can be decrypted only by the secret *private* key. In addition, your private key can be used to "sign" a message you send in such a way that anyone with your matching public key can confirm that *you* are the one who sent the message. This clever arrangement is what enables different forms of public-key cryptography to be used for secure email (using PGP, GnuPG, and S/MIME), iMessage, secure websites (using HTTPS), and more.

WebAuthn applies the same concept to authentication for websites. When you create an account at a site that uses WebAuthn, you don't send them a password; instead, your device creates a new key pair associated uniquely with that particular site, and sends the site the public key. That's the only thing stored on their server, and remember, it's *public*. So it doesn't matter if that key is leaked or stolen, because it was never supposed to be private in the first place! That means your credentials can't be compromised by someone hacking a website. And a malicious party can't use a stolen private key from one site to log you in on a different site, either, because each site has a different key pair and your private key for the second site wouldn't match.

When it comes time to log in, the site sends you some data encrypted with your public key. Your device uses your private key to decrypt it and then returns a signed reply that the server can validate with your public key. If that process succeeds, you're logged in. If not, not. Simple as that.

The security of this scheme depends on whatever's on your device remaining private. And we wouldn't want someone who stole (or borrowed) your device to be able to log in anywhere with no proof that they're actually you! So, the system has an additional layer. Before your device can use your private key to answer the message from the server, you have to prove to the device that you're you, which can be

done via a fingerprint scan (for example, on a Mac with a Touch ID sensor), by typing your device's login password or passcode, by scanning your face (on a suitably equipped device), or in some other fashion. Behind the scenes, your private key is itself encrypted—and it's unlocked only when you authenticate in whichever way your device supports. So, even if someone stole all the data from your device, they'd be unable to decrypt it (and thus log in to passkey-protected sites) without your fingerprint, face, login password, or passcode.

And that makes passkeys safe from *phishing*, in which someone tricks you into revealing your password, perhaps by typing into a fake website. You don't actually know what your passkey is—you never see it—and even if you did, it wouldn't help anyone else because it's encrypted.

But wait, there's more!

Because passkeys inherently involve at least two factors (see Multi-Factor Authentication)—namely, a thing you have (your device) and either a thing you are (a fingerprint or facial scan) or a thing you know (your login password or device passcode), the use of passkeys eliminates the need for those irritating two-factor codes that are either generated by apps such as Authy or Google Authenticator, or (much less securely) sent to you via SMS.

All this is terrific, or at least has the potential to be, once passkeys are supported more broadly and some other kinks (discussed ahead) are worked out. But as I mentioned earlier in the book, nearly all sites that support passkeys at the moment *still require you to have a password*. And that means, sadly, that all the vulnerabilities associated with that password still exist, too.

Operating System & Browser Integration

Recent versions of macOS, Windows, iPhone, iPad, and Android all have built-in support for passkeys, but the devil is in the details. Not every operating system and browser has the same level of support, and

accessing the same passkey from multiple places is tricky. (I cover Syncing Passkeys Across Devices next.)

Microsoft, unfortunately, lags behind Apple, Google, and numerous third-party password managers in passkey support. The passkeys you can save and use in Windows 10 and 11 are *device-bound*, which means they can't sync across devices (to say nothing of platforms).

Note: Although all Windows 11 computers support passkeys, a small percentage of Windows 10 computers do not. Passkey support in Windows depends on Windows Hello, which in turn requires the device to have a Trusted Platform Module (TPM) chip. Some devices manufactured before 2016 lack this chip.

On every platform, passkeys rely on the system's inherent authentication methods. So, if you have an Apple device that offers Touch ID or Face ID, those methods are used to authenticate passkeys; on a Windows PC, Windows Hello (which can use face or fingerprint recognition, or a numeric PIN) can authenticate you; and on an Android device, a face or fingerprint scan is usually used. If your hardware has none of these features—or if you have them turned off—you can still authenticate with the same password or passcode you use to unlock the device. But, in all cases, your device must be configured to require *some* kind of authentication to unlock; otherwise, passkeys won't work. And in some situations, as I describe ahead, you'll also need to have Bluetooth enabled.

Note: In most cases, a FIDO2-compatible hardware security key can also be used to sign in with a passkey in lieu of other authentication methods; see Physical Keys.

You'll also need a supported browser, which, at this point, is pretty much all of them (including recent versions of Google Chrome and Chromium-based browsers, Edge, Firefox, and Safari).

Where Can You Use Passkeys?

For the time being, the use of passkeys is restricted almost entirely to web browsers. Other apps that ask for passwords can occasionally use passkeys too, but so far, they're few and far between.

As a public service, the developers of 1Password maintain a directory of sites using passkeys—including such notable entries as Best Buy, eBay, Google, LinkedIn, and PayPal. It's not a complete list (and at some point, I'm sure that so many sites will support them that it will be pointless to try listing them all), but it's worth a look.

For any website, implementing passkeys is a nontrivial undertaking. A lot of coding and testing will have to happen on every single site that eventually supports this standard, and that will take time. So, don't expect all your favorite websites to switch to passkeys overnight. Also bear in mind that most sites adding support for passkeys will continue offering old-fashioned passwords as an option, so you can still use the sites with older devices that won't support passkeys.

On that note, I would absolutely like to support passkeys as an option on the Take Control Books website one day. Doing so will require the right tools (and I'm not satisfied with what's currently available) as well as significant time for setup, integration, and testing. I'm sure it will happen, but it will take time.

How to Create and Use a Passkey

Now that you know the background, you're probably wondering how you actually go about creating and using passkeys. The simplest answer I can give right now is: if and when a website offers you the opportunity to create one, follow whatever prompts it presents. Likewise, once you've stored a passkey and are ready to use it, you'll generally enter your username, click a button, and authenticate using biometrics or your device's login password. The exact steps vary from device to device and site to site, though you may find the following pages useful:

- Apple offers instructions for using passkeys on a Mac, iPhone, and iPad.

- Google's Sign in with a passkey instead of a password page includes general instructions for using passkeys with Chrome, Edge, and compatible browsers on all platforms.

Note: At publication time, Microsoft has frustratingly little documentation for consumers (as opposed to enterprise users) about using passkeys, except to log in to Microsoft accounts.

It's also worth mentioning that when sites you already have accounts with begin offering passkey support, it could be anywhere from trivially easy to impossible to add that on to an existing username and password; similarly, once you have a passkey, it may be simple, complex, or impossible to add a password as a backup authentication method.

Syncing Passkeys Across Devices

Besides being supported directly by operating systems and browsers, passkeys can *usually* sync across devices. Indeed, the longer and more technically accurate term for passkeys is "multi-device FIDO credentials." So, once you've set up a passkey for a given site on a device (say, an iPhone), you can use the very same passkey to log in to that site on another device (say, a Mac). That may sound pretty much like what we had already with password managers that sync across devices—and eventually it will be—but for the time being, syncing passkeys involves a number of limitations and gotchas.

Note: Syncing your passkeys between devices means that if you lose one of your devices, you can still log in everywhere. If you lose *all* your devices, you'll have more of a challenge, but generally speaking, if you replace your device and log in with the same Apple, Google, or Microsoft account, your passkeys should sync from the cloud to the new device. If you're an Apple user, you can also use their recovery key, account recovery contact, or manual account recovery.

As of early 2025, the passkey storage and syncing mechanisms built into operating systems and browsers still have limited interoperability, though the situation is gradually improving.

For example, with macOS 14 Sonoma, iOS 17, or iPadOS 17 or later, passkeys stored in iCloud Keychain are accessible from other browsers, as long as the developer has added support (and most have). That means if you previously visited a site in Safari and stored a passkey in iCloud Keychain, then later visit the same site using Firefox, you can use that existing passkey without any fuss. (The reverse is also true: if you save a passkey while using, say, Chrome, you have the option of storing it in iCloud Keychain so that Safari—and other browsers—can access it.)

Note: For another browser to be able to access iCloud Keychain, you must go to System Settings (macOS) or Settings (iOS/iPadOS) > Passwords > Password Options and enable AutoFill Passwords and Passkeys. Note that if you're also using a third-party password manager to manage passkeys, the two techniques can interfere with each other. I recommend choosing one or the other.

That's terrific, but other platforms and browsers still need to catch up. If you use a recent version of Chrome, Edge, or other Chromium-based browser on any platform (including Android and Chrome OS), and you're signed in to your Google account on all devices, your passkeys sync across all those places via Google Password Manager. But, for now, they're siloed within Google's ecosystem; you couldn't use one of those passkeys in, say, Safari on an iPad. And in Windows 10 or 11, Edge can use your Microsoft account to save and sync your passkeys, but those won't be available to other browsers or operating systems.

In October 2024, the FIDO Alliance (which includes many major tech companies) released a draft specification for an interoperability standard that will enable passkeys to be exported and imported easily. Once this is implemented, it will presumably be possible to pass passkey data securely among Apple, Google, and Microsoft systems, as well as third-party password managers.

In the meantime, there are two ways to get cross-ecosystem passkey authentication in most situations:

- **Use a nearby device**. Let's say you have an iPhone or iPad that has stored a passkey for a certain site, and you want to log in to that

site using Edge in Windows 11. Or, you have an Android phone that has stored a passkey, and you want to use that passkey to log in with Safari in macOS. In situations like these that cross ecosystem boundaries, you can use a clever technique involving Bluetooth and a QR code to log in without resorting to a password.

First, the devices involved (such as a computer and a phone) must both have Bluetooth enabled. Then, on the device where you're trying to log in, you should see a button or link to display a QR code. Click or tap that, and when the QR code appears, scan it with your phone (unlocking it in the process if you haven't done so already). The site then uses the passkey on your phone to authenticate you and log you in. (The Bluetooth connection proves that you have the phone with you, and unlocking it proves that you're you.) In some cases, you can even use this process to create a passkey for a new site, storing it directly on your phone instead of on your computer.

- **Use a password manager.** Third-party password managers offer more flexibility here. At publication time, 1Password, Bitwarden, Dashlane, Enpass, Keeper, NordPass, Proton Pass, and RoboForm all support passkeys. Using one of those apps, rather than your operating system, to store and sync your passkeys will give you more flexibility in where and how they're used. (The process for storing and entering passkeys using a password manager is much like that for passwords, except there are fewer steps involved.)

  Indeed, using a password manager to store your passkeys seems to give you the best of all worlds: passwords when you need them, passkeys when they're available, and syncing your credentials across all devices, browsers, and platforms.

  The only downside at present is that there's no way to sync, share, export/import, or otherwise transfer a passkey from one ecosystem to another—for example, if you saved a passkey in Safari using iCloud Keychain, you can't move that passkey into 1Password (or vice versa); until the interoperability standards mentioned above are built into all the various operating systems, browsers, and password managers, you'd have to create an entirely new one.

Appendix A: Use Two-Factor Authentication

Earlier, in Multi-Factor Authentication and Manage Email Options, I said that some companies enable (or require) you to use a combination of factors—things you know, things you have, and things you are—to prove your identity. In the most common implementation, one factor is your password (a thing you know) and the second is an object (a thing you have). The result is two-factor authentication (2FA).

A variation on this process requires your regular password plus a time-based, one-time password (TOTP); that password is generated by an app or sent to you via SMS or email. Because the TOTP is still, strictly speaking, something you know—and because a single device could unlock your password *and* display your TOTP—systems that rely on this process aren't truly 2FA; rather, they're two-step verification (2SV).

With 2SV enabled, the chance of your account being hacked falls dramatically. Even if someone learns your username and password, they need your phone too. (Of course, if someone steals your phone and knows or can figure out your username, the only barrier left is your password, so it still has to be a good one!)

In this appendix, I want to introduce you to several common 2FA/2SV systems you're likely to run across. I won't give detailed, step-by-step instructions for setting up each one, but I'll describe the process generally, direct you to where you can get specific instructions, and say a few words about how to use each system after you set it up.

Two-Step Verification Basics

Before I talk about specific services, I want to give you an overview of the general process most of them employ, with minor variations.

To set up 2SV, you'll generally follow steps like these:

1. Log in to your account in the usual way, go to a Settings or Security page on the web, and enable two-step verification.
2. Make sure you can obtain a TOTP when necessary:
    - For services that use SMS, enter your phone number. They'll then typically verify your phone number by sending you a code via SMS you have to enter on the site. Only after you've done this will TOTP-by-SMS be available.
    - For services that use email to send your TOTP, enter your email address. Then click the link in the email message you receive shortly to confirm that you do indeed own that email account.
    - For services that use an authenticator app such as Google Authenticator, Authy, or 1Password, use your authenticator app to scan the QR code shown on screen (or, in some cases, type or paste in a secret key) to *seed* the authenticator app with a value that enables it to generate the correct TOTP every 30 seconds. Usually, after you do this, you'll have to confirm that the authenticator app works by entering a TOTP on the Settings page.

Tip: Before you close that browser window or tab, I recommend taking a screenshot of your QR code. Some services let you display it again (for example, if you want to set up a second authenticator app to generate TOTPs for you), whereas others require you to turn 2SV off and back on again to see a QR code—a hassle you can avoid with this extra step.

3. Record any backup or login codes the service presents. Some services offer a list of special one-time-use codes you can keep in a safe place (for example, printed out and stored in your wallet) that you can use in place of a TOTP should you ever find yourself without your phone or authenticator app. Others supply a single emergency backup code you can use to prove your identity and log in if 2SV isn't an option for any reason.

Tip: The best place to securely store the backup or login codes and the screenshot of your QR code is your password manager! I put all mine in 1Password.

After that one-time setup process, you'll be prompted for your TOTP after you enter your username and password. Depending on the service, that prompt may appear only the first time you use a new device or app, once every month or so, or on every login. Check your messages or email, or use your authenticator app, to learn the current TOTP, enter it at the prompt, and you're in.

Use Apple's Enhanced Security Options

Apple offers two different ways to supplement your password:

- **Two-factor authentication:** The modern version of the process prompts you to allow login attempts and then requires you to enter a code; any Apple device already designated as "trusted" can display both the prompt and the code. Two-factor authentication requires 10.11 El Capitan or later, iOS 9 or later, or watchOS 2 or later—and is the *only* method available on 10.13 High Sierra or later, iOS 11 or later, and iPadOS 13 or later. See Use Apple's Two-Factor Authentication, just ahead.
- **Two-step verification:** Apple's previous authentication system, 2SV, supplies the secondary login code via the Find My iPhone app or SMS. It's available only for those using macOS 10.12 Sierra or earlier, or iOS 10 or earlier. See Use Apple's Two-Step Verification, later.

Tip: You can use one of these systems or the other (or neither)—but not both, even if your operating system supports either version. If all your devices support it, 2FA is definitely the better choice. And, if you want to use Auto Unlock with your Apple Watch, you'll need to enable 2FA first.

The use of 2FA or 2SV is optional, but apart from the enhanced security they offer, there are additional reasons to consider using one of them:

- Setting up either a 2SV or 2FA system eliminates the security questions from your Apple Account—an attacker can no longer use them to break in to your account. So that's a lovely bonus.
- You'll need to have one of these systems activated in order to Use App-Specific Passwords, and Apple now *requires* app-specific passwords for third-party apps (such as Outlook and BusyCal) that connect to your iCloud account.

Use Apple's Two-Factor Authentication

Two-factor authentication for your Apple Account is both simpler and more secure than the older 2SV system I describe later. It's available to all iCloud members who have at least one device running iOS 9 or later, iPadOS 13 or later, or 10.11 El Capitan or later (although you'll have better results if *all* your devices are running compatible operating systems).

With this feature enabled, here's what you'll see:

- The device you use to enable 2FA becomes your first *trusted device* (see Trusted Devices). That means you'll use this device (along with your password) to authenticate on your next device, which then also becomes trusted. You can have several trusted devices.
- When you sign in on a device that isn't yet trusted, you enter your username and password as usual, but then two things happen:
    - You must click or tap an Allow button that appears automatically on all your existing trusted devices and shows the geographic location of the new device that is trying to sign in. (Once you click or tap it on one device, the alert disappears on the rest.)
    - On the new device, you must enter a 6-digit TOTP (which Apple calls a "verification code") that's displayed automatically on all your existing trusted devices. (If you don't have access to a

trusted device, you can receive your TOTP via SMS or a phone call instead.)

- Apps that access your iCloud account must Use App-Specific Passwords, just as with 2SV.
- If you try to sign in to your iCloud account using an older device that isn't running at least iOS 9 or OS X 10.11, you must get a TOTP from a trusted device (or via SMS or phone call) and then *append* that to your password when you enter it. For example, if your ordinary iCloud password is `x?u3[iFirHKL6XTG` and your verification code is `295634`, you would enter `x?u3[iFirHKL6XTG295634` as your password.

In practice, I find Apple's 2FA to be smoother and less cumbersome to use than 2SV, but the setup process is not necessarily intuitive. It's especially weird for people switching manually from the old 2SV system, as they must first turn it off and jump through the additional hoop of setting up security questions (which will become irrelevant a few minutes later, once 2FA is turned on).

To turn off 2SV, if it was previously enabled:

1. Go to appleid.apple.com in a browser, sign in, and in the Security section (where it should say "Two-Step Verification On"), click Edit.
2. Click "Turn off two-step verification" and follow the prompts; note that you must select and answer three new security questions (all of which must have different answers) and supply your date of birth.

To enable 2FA, use a Mac running 10.11 or later, an iPhone running iOS 9 or later, or an iPad:

1. On a Mac running Sequoia or later, go to System Settings > *Account Name* > Sign-In & Security, enter your password if prompted, and click Turn on Two-Factor Authentication. For earlier versions of macOS, go to System Settings/System Preferences > Apple ID > Password & Security (Catalina through Sonoma) or System Preferences > iCloud > Account Details (Mojave). Click Security, enter your password, and click Continue. (In some cases, you may have to

repeat some or all of this process.) Then click Turn On Two-Factor Authentication.

Or, on an iPhone or iPad, go to Settings > *Your Name* > Password & Security. Then tap Turn On Two-Factor Authentication.

2. Follow the prompts. These will include answering two of your three security questions and supplying your date of birth and a phone number; you can select either "Text message" (for SMS) or "Phone call" (for a voice call).

Once you complete these steps, 2FA is on, which means that you may be prompted for your password and TOTP (or, in some cases, an app-specific password) on devices already signed in to your iCloud account.

Note: Be sure to check that your various third-party Apple Account–enabled apps are now working properly, since they may not prompt you to immediately set up app-specific passwords. See Use App-Specific Passwords, ahead.

When you sign in on a new device (or in a browser—for example, if you later visit appleid.apple.com again), the process is as follows:

1. Enter your Apple Account username and password as usual when prompted.
2. An alert (**Figure 4**) appears on every already-trusted device. If the presence and location of the sign-in attempt (which is based on the device's IP address) is what you were expecting, click or tap Allow.

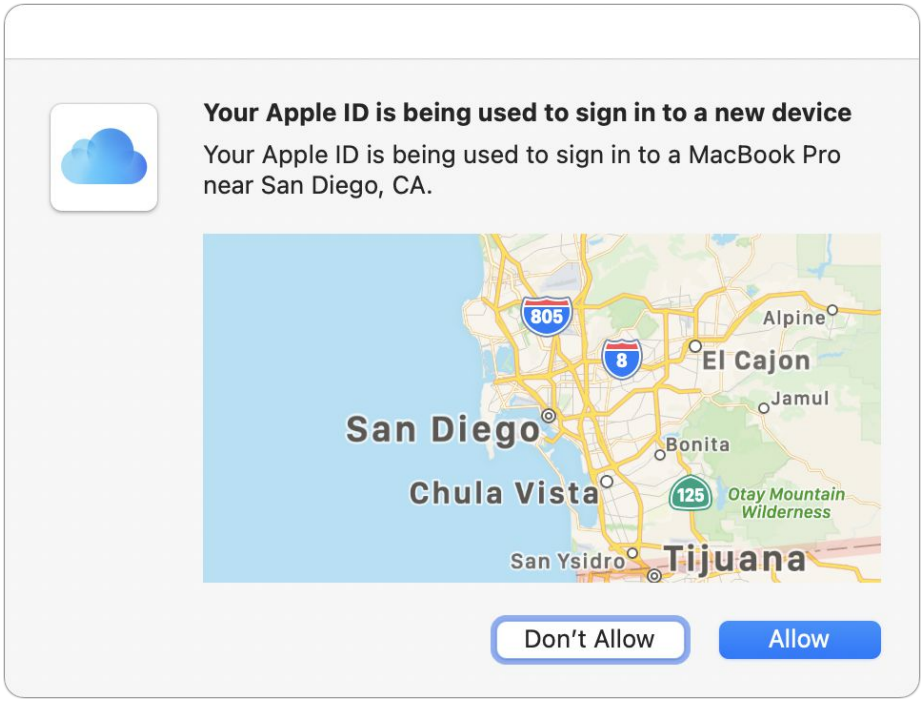


Figure 4: If the area shown on this Mac corresponds to the location where your new device is trying to sign in, click Allow.

3. A 6-digit TOTP (**Figure 5**) appears on the device on which you clicked or tapped Allow.

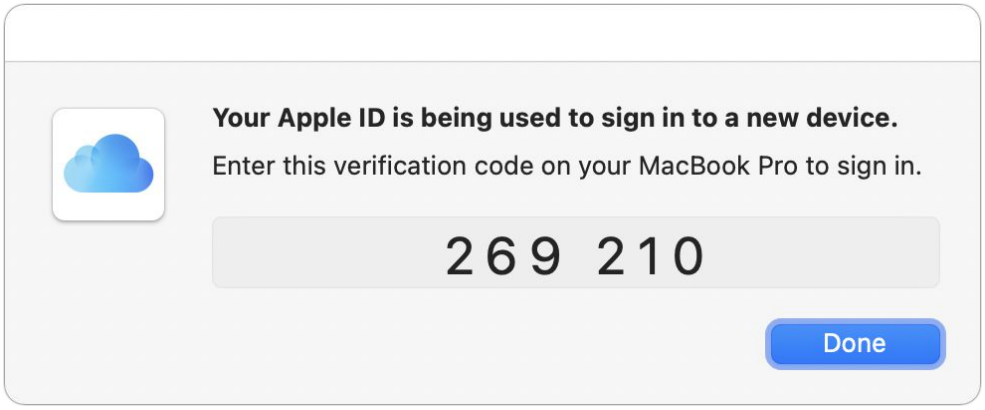


Figure 5: After you click or tap Allow, you see the verification code that you can enter on the device on which you're signing in.

4. On the device you're using to sign in, enter the TOTP (**Figure 6**).

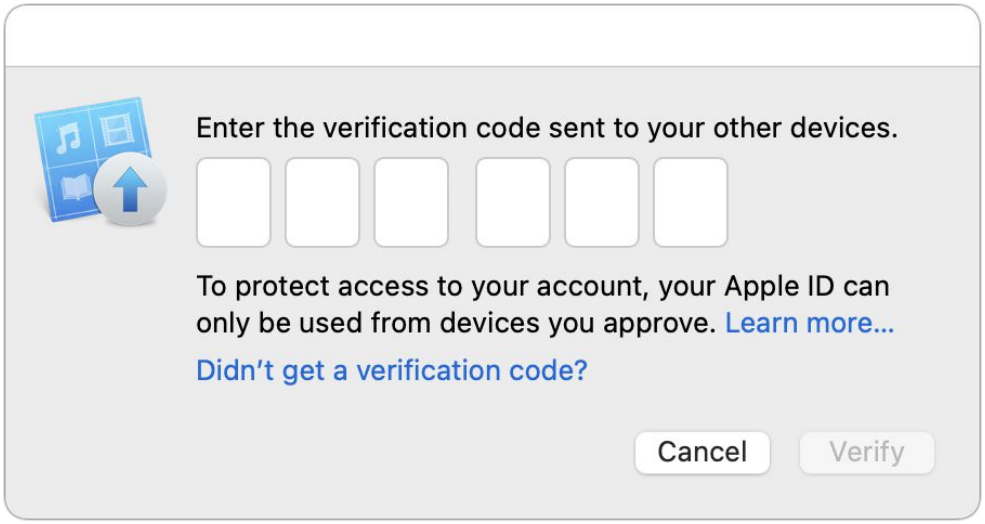


Figure 6: Enter the TOTP from one of your other devices here. In most cases, you won't even need to click Verify.

After you complete this step, your new device becomes trusted.

To learn more about Apple's 2FA system, read:

- Apple's support articles Two-factor authentication for Apple Account and Availability of two-factor authentication for Apple Account
- *Take Control of Your Apple Account*, by Glenn Fleishman, which has detailed, practical steps for setting up, using, and troubleshooting Apple's 2FA (and 2SV, described next), along with a look at what to do if you lose a device or forget your password
- My book *Take Control of iCloud*

Use Apple's Two-Step Verification

Two-step verification—intended for devices running operating systems older than iOS 9 or 10.11 El Capitan—works like this: After you enable it, when you try to perform certain tasks (such as logging in to the iCloud website, changing your password, or making a purchase on a new device), you enter your username and password. Then you're prompted to supply a numeric code that's sent to an iPhone or via SMS to an ordinary cell phone. Only after you enter this code are you granted access.

You won't have to go through these steps very often. With 2SV, you'll be asked for the TOTP when you:

- Sign in to appleid.apple.com to manage your account
- Sign in to iCloud for the first time on a new Mac or iOS device
- Sign in to your account on the iCloud website (although you can check Remember This Browser to avoid being prompted for verification in the future when using the same browser on the same device, and you can use Find My *Device* without verifying your identity)
- Make a purchase from iTunes, Apple Books, or the App Store for the first time on a new Mac or iOS device
- Contact Apple for support with your Apple Account

In any of these situations, you'll be prompted to choose a particular device or SMS number to use for receiving the one-time password. Once you enter it, you can use the app, site, or service as usual.

In addition, 2SV, once enabled, takes the place of security questions when you have to reset a lost or forgotten password. That's a very good thing.

You can find additional details about how Apple's 2SV works and how to set it up in the support article Two-step verification for Apple Account.

Use App-Specific Passwords

Enabling Apple's 2FA or 2SV also activates another security feature: app-specific passwords. This feature applies to all third-party apps (such as Outlook, Thunderbird, BusyCal, and BusyContacts) that access your iCloud account. It does not apply to Apple apps (such as Mail, Safari, and Find My Friends), although previously it applied to FaceTime, Game Center, and iMessage accounts in Messages.

With app-specific passwords, Apple generates a special, unique password for each of the affected apps—*your ordinary iCloud password no*

longer works for these apps when 2SV or 2FA is active. You can create app-specific passwords only after going through the 2SV or 2FA process, so you're still protected by both steps—but once you've done this for a given app on a given device, you're never prompted to do so again (for that app on that device), unless you change your security settings.

To set up an app-specific password:

1. Open a third-party app that connects to your iCloud account. (If you're doing this for the first time after enabling two-step verification or two-factor authentication, you may see an error message stating that your password wasn't accepted.)
2. Locate the app's settings for username and password (often in the Settings/Preferences window). Leave the window open.
3. Visit appleid.apple.com in your browser, sign in, and verify your identity.
4. In the Security section, under "App-Specific Passwords," click or tap Generate Password.
5. Type a name for the app (such as `BusyCal iMac`) and click Create.

Tip: It's safest to create a separate password for each app on each device rather than to reuse a password in the same app on multiple devices. Thus, I suggest including the device in the name you type here, so it's easier to identify if you ever need to revoke the password (as I explain just ahead).

6. The new password appears on screen. Copy and paste it (or type it) into the window you opened in Step 2.

Tip: Apple provides no way to view your app-specific passwords after the fact. (You can see their *names,* but not the actual passwords.) So you might want to record the password in a safe place, such as your password manager, to avoid the extra step of regenerating a new password for this app if you need it again later.

If a device is lost or stolen, you may later want to revoke an app's password. Doing so prevents that app, on that device, from accessing your iCloud account. Follow the same steps, but when you get to Step 4, instead click Edit and then click View History. You can then click the X icon to revoke a single password or Revoke All to revoke them all.

Note: See the Apple support article Sign in to apps with your Apple Account using app-specific passwords for further details.

Use Dropbox's Two-Step Verification

If you activate Dropbox's 2SV, then each time you sign in to Dropbox on a new device, you must supply not only the password but also a TOTP obtained from an authenticator app such as Google Authenticator, Authy, or 1Password. Once you authorize an app on any platform, that authorization sticks until you revoke it—you won't have to repeat this every time you use the app. And, you can optionally click a "Trust this computer" checkbox when signing in to Dropbox in a Mac or PC browser so you won't have to repeat this process every time you use the Dropbox website.

Note: Any device or app that you've authorized with conventional password-only authentication remains authorized when you switch to two-step verification, and any browser session where you're logged in remains active. This change applies only to *future* sign-ins.

The Dropbox website provides instructions for enabling two-step verification, and I won't repeat them here. However, one thing that the instructions don't make clear is that although you have to choose between using an authenticator app and an SMS message for authentication, there's a way to keep both options available. To do this, choose Use Mobile App when prompted to pick a method. Later you'll be asked if you *also* want to add an SMS-capable phone number as a backup. So, do that too. Then, if you're ever in a situation where you're signing in to Dropbox and you prefer to use SMS rather than a mobile app, click the "I lost my phone" link, and you can then receive your

code by SMS (without in any way affecting your ability to use a mobile app in the future).

Note: If you really have lost your phone—the same one you use for SMS authentication—then of course you won't be able to access either your authentication app or your SMS messages. But Dropbox lets you add a second, backup SMS number if you like, for just such situations.

In any case, when you sign up for 2SV, Dropbox provides you with ten 8-character emergency backup codes that you should keep in a safe place. If you ever find yourself without access to your mobile authenticator app or SMS, one of these codes will serve as a secondary password, enabling you to sign in.

Use Facebook's Two-Step Verification

Facebook refers to its 2SV system as two-factor authentication (which, as I've pointed out, is not quite accurate); formerly, it was known as "login approvals." Like other services I discuss in this appendix, Facebook lets you receive TOTPs via SMS or a third-party authenticator app. But the preferred (and arguably easiest) way to get them is to use the Code Generator feature built into the Facebook mobile app itself.

Two-step verification is easy to set up. Once you've done so, you can optionally print out login codes that you can use if your phone, authenticator app, or mobile Facebook app are unavailable. For complete details, see these Facebook support articles:

- What is two-factor authentication and how does it work?
- What is Code Generator and how does it work?
- How do I use an authentication app for two-factor authentication?
- How do I get Facebook recovery login codes to use when I don't have my phone?

Use Google's Two-Step Verification

If you use Google's services (such as Gmail, Google Docs, Google Voice, and about 937 others), you can optionally enable 2-Step Verification, which uses either SMS, an automated phone call, or an authenticator app to deliver a one-time password when you need to log in. Yet another option is Google prompts, which let you tap a single button on a trusted mobile device. You can tell Google to remember you on any given device or app for 30 days, during which time you can log in with just a username and password.

Note: Google also lets you use a U2F USB security key, such as a YubiKey, for 2SV—which effectively turns it into 2FA.

During the setup process, Google supplies you with backup codes you can use if your ordinary channels of 2SV aren't available. You can also specify a backup phone number at which you can receive TOTPs if your regular phone can't be used.

So how does this work for apps that access Google services directly but don't have a way to ask you for that TOTP—such as the email client on your cell phone? To deal with situations like these, Google offers app-specific passwords much like the ones Apple uses in similar situations. To use one of these, you'll log in to the Google website with 2SV. Then you'll go to a special page where you can generate a unique password for each app, on each device, that has to access Google directly. Go back to the settings on each device, pop in the unique password (instead of your regular Google password), and you're done—access happens normally after that.

Google provides thorough, clear instructions for two-step verification, so rather than reiterate it all here, I'll refer you directly to the source:

- For users with ordinary Google accounts, read about setting it up and using it in Protect your account with 2-Step Verification.
- For Google Workspace administrators, read setup and usage instructions in Protect your business with 2-Step Verification; for

Google Workspace users, read Protect your account with 2-Step Verification.

- For information on using app-specific passwords, see Sign in with App Passwords.

Use Microsoft's Two-Step Verification

If you use Hotmail, Microsoft 365, OneDrive, or any of numerous other Microsoft services, you have a Microsoft account. Like other accounts that protect valuable resources, it can be enhanced with two-step verification. It requires the use of an TOTP, generated by an authenticator app, the first time you log in with a particular service on any given device. You can optionally check a box to indicate that the device should be trusted in the future, so you won't be prompted for 2SV again on that device.

To learn more details and the exact steps to follow, read Microsoft's How to use two-step verification with your Microsoft account page. Once you've set up 2SV, you'll need to create and enter app-specific passwords for most apps and devices that log in to your Microsoft account; Microsoft explains the process in How to get and use app passwords.

Appendix B: Help Your Uncle with His Passwords

As much as it pains me to admit this, there are people who will listen patiently to a sober description of the problems with passwords and my simple strategy to overcome them, and say, "Yeah, no, sorry. I'm Just Not Going to Do That."

You, of course, aren't one of those people. You eat your vegetables, work out, drive safely, and have amazing passwords. But you have a friend—or perhaps a much older, much younger, or less technologically sophisticated family member—who's too busy, too set in their ways, or for some other reason unwilling or unable to follow my strategy. You want them to be safe, but no matter how much sense it might make, you know they're not going to go for the plan in this book. What to do?

In this appendix, I offer a few suggestions for helping such a person, organized by potential areas of compromise. You may not be able to break every bad habit, but you can perhaps meet your "uncle" halfway and make him that much more secure.

Password Manager Compromises

Your uncle may refuse to use a password manager, considering it too inconvenient or too difficult. Or, he may have so few passwords that a password manager would be overkill. Either way, if a password manager is out of the question, you can make a couple of suggestions:

- For a modest number of passwords, a piece of paper can be a completely adequate password manager. And, if your uncle does all his computing from his home in the country, the likelihood of someone finding and stealing that paper is small. But tell your uncle that keeping it out of sight is still best.

- Even without a password manager, a password *generator* will help your uncle come up with better passwords. He can find lots of free web-based password generators online—for example:
  - 1Password Strong Password Generator
  - Strong Password Generator
  - Random Password Generator

Password Reuse Compromises

I've explained the dangers of using the same password in multiple places and repeated my warning numerous times. But some people—especially if they can't or won't use a password manager—can't accept the notion of having lots of different passwords. "It's too much to remember!" If you encounter such a person, try these compromises:

- If you're going to reuse a password, at least make it a great password (see All About Entropy).
- Ask if your uncle might be willing to remember two (or even three or four) great passwords, and alternate among them for various sites. That'll require either reminder notes or the extra step of guessing on occasion, but at least it'll contain the risk a bit.
- Alternatively, perhaps your uncle would be willing to memorize a single great password like `vGq9&nn3c3#b` and vary a portion of it for each site. For example, maybe the second character, `G`, stands for "Google" but when your uncle logs in to Amazon.com the password becomes `vAq9&nn3c3#b` and at PayPal he uses `vPq9&nn3c3#b`.

Note: If you do a pattern-based substitution like this—and remember, this is only for your uncle, not for you!—don't be blatant (as in `Goog&nn3c3#b` and `Amaz&nn3c3#b`), because if anyone learned one of those passwords it would be obvious what all the rest were!

- Suggest that your uncle set up an extra, non-public email address to use as his standard username (as discussed in Manage Email Op-

tions). If an attacker doesn't know his username, his reused password is less vulnerable.

Password Complexity Compromises

Perhaps your uncle balks at the idea of long, random (or random-looking) passwords. What can you suggest? Here are some ideas:

- Explain that he can have an extremely strong password made of all English words—as long as there are enough total letters and the words are randomly chosen. Don't let him use `correct horse battery staple`, but help him come up with a comparable phrase (or several) that's a bit longer—think five words. If you do this…
  - Suggest including an uppercase letter in each word—not necessarily at the beginning (`icinG enougH friendlY skunkS`).
  - Consider skipping the spaces between words, because a lot of sites won't take them. Instead, see if your uncle might be willing to type a number and/or a punctuation character between words. For example, a password like `icinG.1enougH.2friendlY.3skunkS` is not half bad. Another benefit of doing it this way is that when your uncle encounters a site that won't accept such long passwords, he can enter just the first however many characters and still get the benefits of a large character set.
  - Misspelling words adds entropy, and egregious misspellings are better than simple ones. So a password like `Phraindle-Scuhnx` is a huge improvement over `Icing-Enough-Friendly-Skunks`.
- Urge your uncle to make his existing passwords *just one character longer*. Each character increases password strength exponentially!
- Prepare a little speech along the lines of, "Uncle Alfonso, if pirates wipe out the $234 in your bank account, don't come running to me!" Repeat that a few times. And then, right when your uncle is starting to lose his temper, explain that you just gave him his new ultra-strong password: `UA,ipwot$234iyba,dcrtm!`

Appendix C: Calculate Password Strength

Lots of websites and password generators have little meters that claim to tell you how strong a password is—they want you to keep adding characters until the bar is long enough or turns green or whatever. The problem is, each meter uses its own method to estimate password strength. The results vary wildly, and a tool may give you a false sense of security by suggesting that your password is stronger than it really is. (For more on this problem, read Does your password pass muster? Password strength meters not all created equal at ScienceDaily.)

Although not perfect, the best online password meter I've found (and one that ScienceDaily likes too) is an open-source tool from Dropbox called zxcvbn. Not only will it tell you a password's strength and the estimated time to crack it, it will also point out specific areas of weakness (such as dictionary words and patterns in the password). And don't worry, it does all this safely within your browser—it doesn't transmit your passwords over the internet.

We'll come Back to zxcvbn in a moment. First, we need to cover a little bit of math. Honestly, it's *very* little, but I want to explain briefly how one goes about calculating password strength mathematically—and in particular, how to arrive at this mysterious concept of *entropy* (a password's resistance to being guessed), measured in *bits*. It's a calculation you can do easily, all by yourself, with a calculator or a web browser—and it lets you prove to yourself how strong any given password is. (It's also subject to a lot of qualifications and caveats, as we'll soon see, but first things first.)

An Aside: Doing Math with Google

Alert readers may have noticed that in the foregoing discussion I left out one teensy detail, which is how one actually goes about calculating

a base-2 logarithm. Well, there are lots of ways to do it. Here are three of them:

- **Scientific calculator, one-step method:** Some scientific calculators can calculate base-2 logarithms with a $\log_2$ key—so enter the maximum number of guesses and then press that key. Other calculators (including the Mac's built-in calculator in Scientific mode—choose View > Scientific) offer only base-10 logarithms. In which case…
- **Scientific calculator, two-step method:** If your calculator offers only base-10 logarithms ($\log_{10}$ or just plain $\log$), no worries. You can calculate a base-2 logarithm by taking the base-10 logarithm and then dividing it by log(2). In other words, $\log_2(x) = \log_{10}(x) / \log_{10}(2)$. No calculator at all? Move on to…
- **Google:** Did you know Google has a built-in scientific calculator? It does, and it's really good. Just go to google.com, and instead of typing in a search term, type a formula, for example: `log2(95^12)`

  Press Return, and you'll get the answer instantly (**Figure 7**).

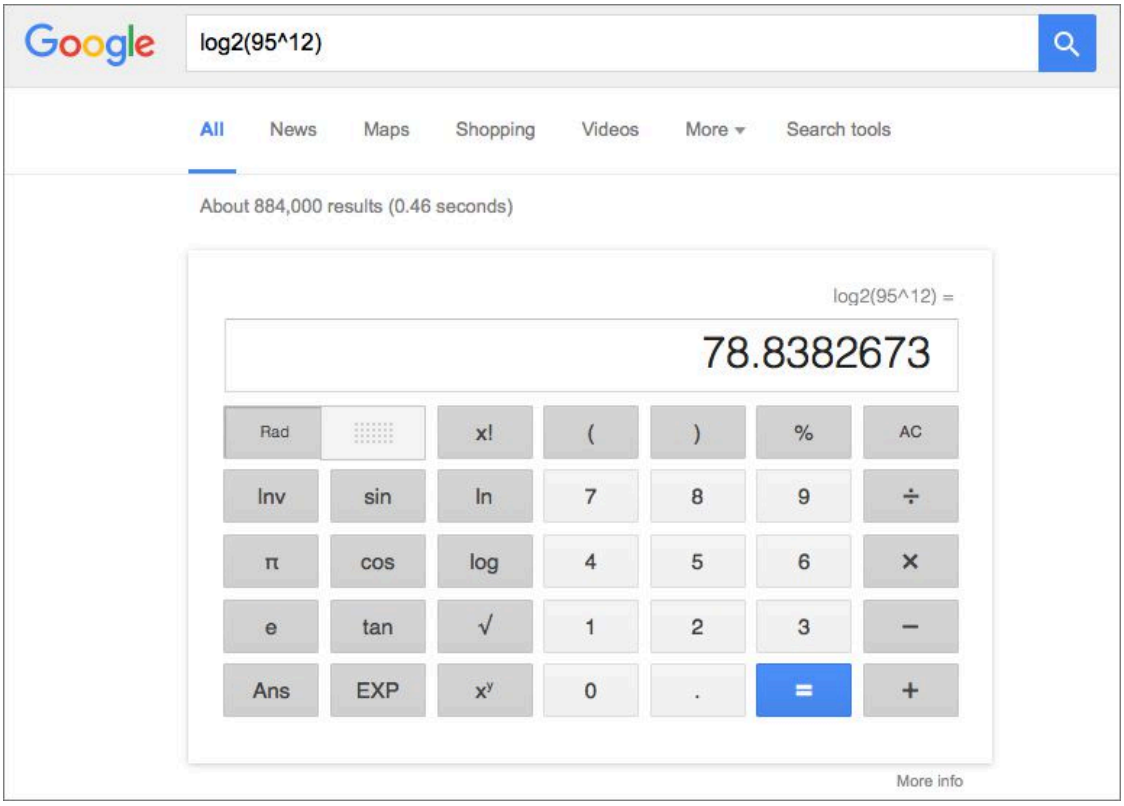


Figure 7: You can calculate a password's entropy (or just about anything else) using Google's built-in scientific calculator.

Notice two things about that formula:

- Google doesn't use subscripts, so if you want a base-2 (or base-whatever) logarithm, you just type that number after `log`.
- Google doesn't use superscripts, so to raise `x` to the power of `y`, you use a caret (`^`) in between them—`x^y` is the same as x^y.

In summary, the easiest way for most of us to compute a password's entropy in bits is to go to google.com and enter `log2(x^y)`, where `x` is replaced with the number of possible characters in each slot, and `y` is replaced with the total number of characters in the password.

The Entropy Formula

If you're mathematically inclined, you may be able to make sense of the formula for entropy without an extensive explanation. Here it is:

$$\log_2(\text{possible characters}^{\text{length}})$$

If that doesn't make sense to you, don't worry; it didn't make sense to me either before I started writing this appendix! But I think I can unpack it in a way that anyone with basic knowledge of algebra can understand. So if that looks like so much gibberish to you, read on!

If cryptographers and mathematicians were a bit more inclined to think like the rest of us, they'd tell us how hard it is to guess any given password using an ordinary number—a number that represents the total *search space* for any given password, or the maximum number of guesses that could be required to match it, if you had to try every possible combination of characters. This number would probably be pretty big, but at least it would be just that—a straightforward number.

Note: I specify "maximum number of guesses" because if you're running through every possible combination of characters, it's highly unlikely that the particular password you're searching for will be the very last one you check. On average, you'll find it about halfway through your search.

For example, if you have a password made up of all lowercase letters, and it has 6 characters, then the total number of possibilities is 26 × 26 × 26 × 26 × 26 × 26, or 26^6, or 308,915,776. Call it 300 million and change. That's the maximum number of guesses it could take to find a password meeting those criteria.

But, since mathematicians make things "simpler" by making them harder, they don't just toss out a number like 300 million. They perform a calculation to reduce very big numbers like 308,915,776 to smaller numbers that are—even I, as a layman, must admit—easier to work with, especially when the number of possible passwords starts getting into the billions, trillions, and beyond.

To do this, they use a mathematical operation called a *logarithm*. A logarithm is the reverse of exponentiation. So, if 10^3 is 1,000, then log(1,000)=3. (And 3, you'll notice, is a smaller and more convenient number to work with than 1,000!) That assumes base 10—which is to say, the number that you're raising to some power is 10. What if you're raising a number other than 10 to some power? Like, say, 2^3, which

equals 8? In that case, you'd want the base-2 logarithm, or $\log_2$. (Similarly, if you're exponentiating some other number, that other number becomes the base.) So, $\log_2(8)$ also equals 3, because 3 is the number to which, if you raise 2, you get 8.

So far so good? I hope so, because that's pretty much it. For the purposes of calculating password entropy, base-2 logarithms are exactly what we need, because that's where the "bits" come from. If I tell you that a certain password has an entropy of 16 bits, that means it would take a maximum of 2^{16} (or 65,536) guesses to find it. Back to our earlier example, the base-2 logarithm of 26^6 (or 308,915,776) is roughly 28, which is another way of saying that 26^6 is the same as 2^{28} (I'm rounding a bit to keep the numbers easier to work with).

Conversely, if you know the maximum number of guesses that will be needed, you can determine the number of entropy bits a password has with this formula:

$$\log_2(\text{max number of guesses})$$

And where does that maximum number of guesses come from? It's the number of possible characters that could be in each slot raised to the power of the number of characters in the password. To our earlier example, if you're using only lowercase letters, then there are 26 possible characters in each slot, and then you'd raise that to the power of the total number of characters in your password (say, 6).

If, on the other hand, you construct your password using a combination of lowercase and uppercase letters, digits, and any of the punctuation keys on a standard keyboard, that's 95 possible characters in each slot, so a 6-character password would have 95^6 possible combinations, or 735,091,890,625 (about 735 billion). In any case:

$$\text{max number of guesses} = \text{possible characters}^{\text{length}}$$

Put it all together and you end up with the formula for a password's entropy presented at the beginning of this discussion:

$$\log_2(\text{possible characters}^{\text{length}})$$

Plug in the size of the character set and the number of characters, and you have an entropy measurement. For example, a 12-character password constructed from any of the 95 available characters on your keyboard would have an entropy of $\log_2(95^{12})$, or 78.8382673 bits. (Which, by the way, is a pretty strong password!)

Why That Entropy Formula Is Wrong

Now that I've shown you a tidy way to calculate entropy, it is my sad duty to explain why the number of bits you got may be way too high, because when it comes to passwords, nothing is as simple as it looks.

The working assumption with a standard entropy calculation is that whatever combination of characters makes up your password, it was chosen *randomly*. That is, your password is no more or less likely than any other password that has the same number of characters and is constructed using the same set of possible characters.

But that assumption is usually wrong, which in turn means that a calculated entropy value is likely to be far too optimistic. Any patterns used in the generation of your password—even if you were consciously unaware of them—increase the likelihood of guessing that password and thereby reduce its *effective* entropy.

To illustrate how this works, suppose I give you a list:

```
w>AXYmPTsig2Fk&4

gUYMtQ>a79F+cgJa

ct(ZL7kFj.Eb3Rks

MB7cJTvxRAR9Eb;%
```

Those are four fine-looking passwords, each one 16 characters long, randomly generated from a large set of possible characters. On paper, each one should have an entropy of 105 bits—not too shabby. But now I tell you, when selecting a password for a new account, that you must choose one of those four passwords—you can't use anything else.

In other words, your password generation method is to make a one-in-four choice. It's exactly as secure as picking a random number from 1 to 4. There are four options, so the effective entropy is 2^2, or 2 bits. And 2, I'm afraid, is a considerably smaller number than 105!

Your calculator (or whatever password strength meter you plug one of those passwords into) won't know how you went about generating it. It will assume each character was chosen randomly. But since the choice wasn't random at all (unless you want to count one-in-four odds as random), your calculator has misled you into thinking your password is stronger than it really is.

All that to say: it is not only the inherent composition of a password that determines its strength, but rather the method used to generate it. If that method is entirely random, then you can rely on your entropy calculations. If not, not.

The reason your method of generation is so crucial is that hackers are incredibly smart, and the cracking tools they use are unbelievably sophisticated. They'll find almost any pattern or shortcut that may have been used in selecting a password, and that will enable them to jump to the correct answer without trying every possible combination.

For example, I spoke earlier of so-called dictionary attacks, in which an attacker starts by trying every word in the dictionary (all in the blink of an eye) before moving on to more sophisticated patterns and then to random combinations. Let's use, as an illustration, Merriam-Webster's Collegiate Dictionary, 11th edition, which contains about 70,000 definitions. If you compute the base-2 logarithm of 70,000, you'll find that it's just a hair over 16. That means if your password is in that dictionary, and the attacker tries every word in that dictionary, then no matter how long or obscure that word is, its effective entropy is only 16 bits. It will take an attacker a maximum of 70,000 guesses—and, on average, half that—to find your password. All that can happen before you say "ouch."

In total, the English language has around half a million words (about 19 bits of entropy each)—that's what you'll find in an unabridged dictionary. The dictionary lists actually used in sophisticated password

hacking contain far more terms—words in other languages, proper names, and of course the most common passwords people have been found to use in real life. Plus common mutations and patterns based on those terms. Maybe there are tens of millions of entries altogether (let's say, conservatively, 23 bits of entropy for each one), and every one of them can be checked in a flash.

Although this makes the math quite messy, none of this should discourage you, because the bottom line remains the same: choose a *long* password and choose it *randomly*, and you'll be fine. (And, to reiterate, as long as your password is truly generated randomly, all the entropy calculations from earlier in this appendix are still valid!)

Back to zxcvbn

I told you that we'd get back to the zxcvbn password strength estimator, and here we are. An older version of zxcvbn showed each password's entropy in bits, and that was what I used for all my examples in the first edition of this book. Since then, the developers have improved zxcvbn, but those improvements have a couple of significant implications I want to point out:

- Its password strength calculations now take into account a much larger dictionary and a number of additional patterns. That means many passwords that the old version of zxcvbn rated as strong are now considered much weaker. (That's only reasonable considering the improvements in password cracking. In fact, I daresay zxcvbn is still too optimistic by an order of magnitude, because professional crackers use far bigger dictionaries than zxcvbn does.)
- Although zxcvbn still calculates *strength*, the official demo page linked above doesn't directly report a password's *entropy*. It's a fair change because what it reported previously wasn't truly entropy in the strictest sense but rather a $\log_2$ calculation based on the number of guesses zxcvbn thought one might need to crack a given password, after taking into account the dictionary and other patterns.

Note: To see what zxcvbn's entropy calculation *would have been* with the updated algorithm, find the line labeled `guesses_log10` after checking a password, and calculate `log2(10^guesses_log10)`. That's how I got the values in All About Entropy, and it's why (as I said there) they were based on a relatively small dictionary. Even easier: see this page by Stephen Wetzel, which shows old and new versions of zxcvbn side by side, with entropy calculations in bits for each.

Password Strength Summary

Putting this all together, let me summarize the main things you need to know about password strength:

- Password entropy is easy to calculate using The Entropy Formula.
- Entropy calculations are reliable *only* for *randomly* created passwords. (So, make sure your passwords are all random!)
- Increasing a password's length by one character increases entropy more than widening the set of possible characters by 10. For example, $\log_2(84^{10}) < \log_2(95^{10}) < \log_2(84^{11})$. Therefore, making a password longer is the easiest way to make it stronger.

For Further Reading

If you find password strength as fascinating as I do, you may want to read more about it. For example:

- Entropy (information theory), Password strength, and Password cracking in Wikipedia
- How 1Password calculates password strength at AgileBits
- Confused about (password) entropy at Stack Exchange
- How do I compute the approximate entropy of a bit string? at Stack Overflow
- Statistics Will Crack Your Password at Praetorian

About This Book

Thank you for purchasing this Take Control book. We hope you find it both useful and enjoyable to read. We welcome your comments.

Ebook Extras

You can access extras related to this ebook on the web. Once you're on the ebook's Take Control Extras page, you can:

- Download any available new version of the ebook for free, or buy a subsequent edition at a discount.
- Access the book in both PDF and EPUB formats. (Learn about reading on mobile devices on our Device Advice page.)
- Read the ebook's blog. You may find new tips or information, as well as a link to an author interview.
- Find out if we have any update plans for the ebook.

If you bought this ebook from the Take Control website, it has been automatically added to your account, where you can download it in other formats and access any future updates.

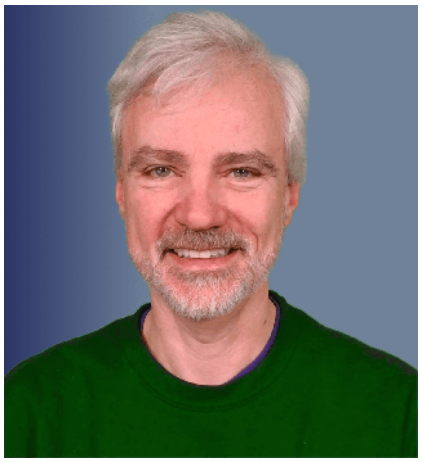
More Take Control Books

This is but one of many Take Control titles, and I'm just one of many Take Control authors! We have books that cover a wide range of technology topics, with emphasis on Macs and other Apple products.

You can buy Take Control books from the Take Control online catalog as well as from venues such as Amazon and the Apple Books Store. But it's a better user experience and our authors earn more when you buy directly from us. Just saying…

Our ebooks are available in two formats, PDF and EPUB, which are viewable on any computer, smartphone, tablet, or e-reader. All are DRM-free.

About the Author and Publisher



Joe Kissell is the author of more than 60 books about technology. As of 2017, he also became the publisher of Take Control Books, when alt concepts—the company he runs along with his wife, Morgen Jahnke—acquired the Take Control series from TidBITS Publishing Inc.'s owners, Adam and Tonya Engst.

Joe was formerly a contributing editor to TidBITS and a senior contributor to Macworld. Before he began writing full-time in 2003, Joe spent nearly eight years managing software development. He holds a bachelor's degree in Philosophy and a master's degree in Linguistics.

In his rare non-work hours, Joe likes to travel, walk, cook, eat, and practice t'ai chi. He and Morgen live in Saskatoon, Saskatchewan, Canada, with their two sons. To contact Joe about this book, send him email and *please* include `Take Control of Your Passwords` in the subject. You can follow him on Mastodon (@joekissell) or Bluesky (@joekissell.com), or visit his personal website, JoeKissell.com.

Credits

- Publisher: Joe Kissell
- Editor: Kelly Turner
- Cover design: Sam Schick of Neversink
- Logo design: Geoff Allen of FUN is OK

Also by Joe Kissell

Click any book title below to add more ebooks to your Take Control collection!

Take Control of 1Password: Use this powerful password manager to create, store, enter, and sync personal data on all your devices.

Take Control of Apple Mail: Learn the ins and outs of Apple's email app in macOS and iOS.

Take Control of Automating Your Mac: Work more efficiently on your Mac with time-saving shortcuts of all kinds.

Take Control of Backing Up Your Mac: Protect your Mac's valuable data from any sort of mishap.

Take Control of DEVONthink 3: Master this powerful information management tool.

Take Control of iCloud: Make the most of Apple's online service for storing, syncing, and sharing data.

Take Control of the Mac Command Line with Terminal: Master your Mac's command-line interface and learn basic Unix skills.

Take Control of Sequoia: Discover what's new in macOS 15, and get all the information you need to upgrade safely.

Take Control of Solving Mac Problems: Learn how to maintain, troubleshoot, and speed up your Mac.

Take Control of Your Digital Legacy: Make sure your important digital information is preserved for future generations.

Take Control of Your Online Privacy: Learn what's private online (not much)—and what to do about it.

Take Control of Your Paperless Office: With your Mac, scanner, and this ebook, you'll finally eliminate the chaos of overflowing paper.

Copyright and Fine Print

Take Control of Your Passwords, Fourth Edition
ISBN: 978-1-990783-30-2
Copyright © 2025, Joe Kissell. All rights reserved.

alt concepts, 419 8B-3110 8th St. East, Saskatoon, SK S7H 0W2 Canada

Why Take Control? We designed Take Control electronic books to help readers regain a measure of control in an oftentimes out-of-control universe. With Take Control, we also work to streamline the publication process so that information about quickly changing technical topics can be published while it's still relevant and accurate.

Our books are DRM-free: This ebook doesn't use digital rights management in any way because DRM makes life harder for everyone. So we ask a favor of our readers. If you want to share your copy of this ebook with a friend, please do so as you would a physical book, meaning that if your friend uses it regularly, they should buy a copy. Your support makes it possible for future Take Control ebooks to hit the internet long before you'd find the same information in a printed book. Plus, if you buy the ebook, you're entitled to any free updates that become available.

Remember the trees! You have our permission to make a single print copy of this ebook for personal use, if you must. Please reference this page if a print service refuses to print the ebook for copyright reasons.

Caveat lector: Although the author and alt concepts have made a reasonable effort to ensure the accuracy of the information herein, they assume no responsibility for errors or omissions. The information in this book is distributed "As Is," without warranty of any kind. Neither alt concepts nor the author shall be liable to any person or entity for any special, indirect, incidental, or consequential damages, including without limitation lost revenues or lost profits, that may result (or that are alleged to result) from the use of these materials. In other words, use this information at your own risk.

It's just a name: Many of the designations in this ebook used to distinguish products and services are claimed as trademarks or service marks. Any trademarks, service marks, product names, or named features that appear in this title are assumed to be the property of their respective owners. All product names and services are used in an editorial fashion only, with no intention of infringement. No such use, or the use of any trade name, is meant to convey endorsement or other affiliation with this title.

We aren't Apple: This title is an independent publication and has not been authorized, sponsored, or otherwise approved by Apple Inc. Because of the nature of this title, it uses terms that are registered trademarks or service marks of Apple Inc. If you're into that sort of thing, you can view a complete list of Apple Inc.'s registered trademarks and service marks.