# A Deep Dive on End-to-End Encryption: How Do Public Key Encryption Systems Work?

19-24 minutes

---

If used correctly, end-to-end encryption can help protect the contents of your messages, text, and even files from being understood by anyone except their intended recipients. It can also be used to prove that a message came from a particular person and has not been altered.

In the past few years, end-to-end encryption tools have become more usable. Secure messaging tools like Signal—for voice calls, video calls, chats and file sharing—are good examples of apps that use end-to-end encryption to encrypt messages between the sender and intended recipient. These tools make messages unreadable to eavesdroppers on the network, as well as to the service providers themselves.

With that said, some implementations of end-to-end encryption can be difficult to understand and use. Before you begin using end-to-end encryption tools, we strongly recommend taking the time to understand the basics of public key cryptography .

The type of encryption we're talking about in this guide, which end-to-end encryption tools rely on, is called public key cryptography, or public key encryption. To read about other types of encryption, check out our What Should I Know About Encryption? guide.

Understanding the underlying principles of public key cryptography will help you to use these tools successfully. There are things that public key cryptography can and can't do, and it's important to understand when and how you might want to use it.

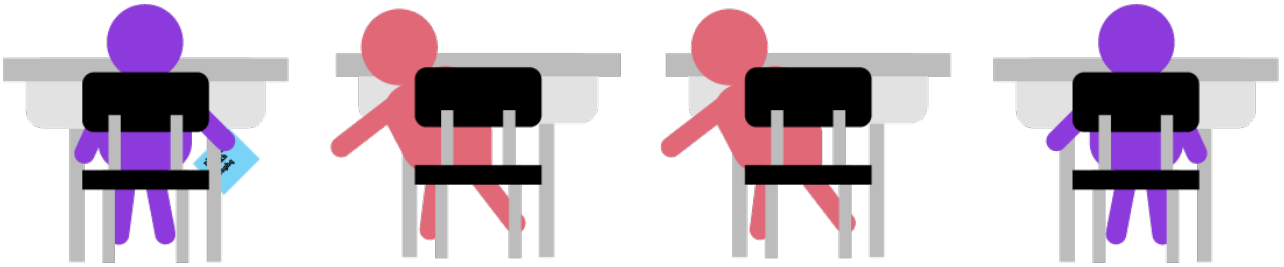
## What Does Encryption Do? anchor link

Here's how encryption works when sending a secret message:

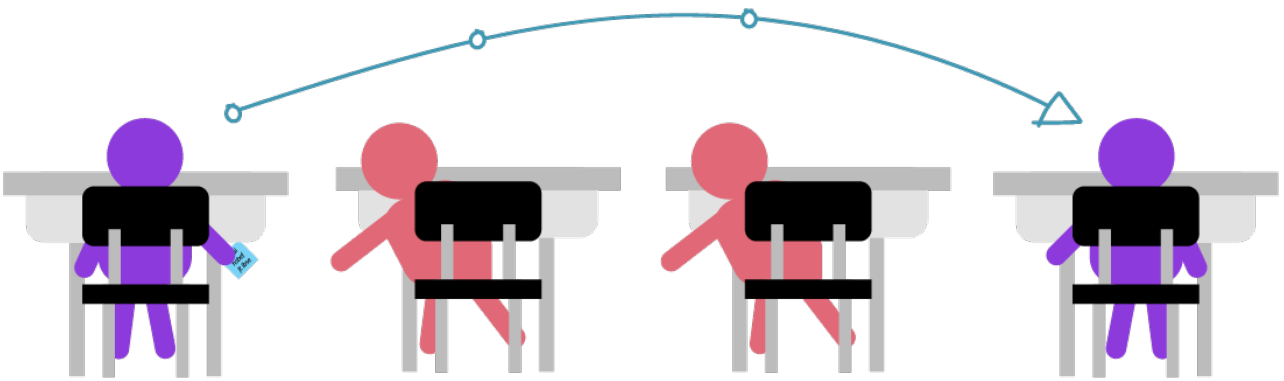
1. A clearly readable message ("hello mum") is encrypted into a scrambled message that is incomprehensible to anyone looking at it ("OhsieW5ge+osh1aehah6").
2. The encrypted message is sent over the Internet, where others see the scrambled message, "OhsieW5ge+osh1aehah6"
3. When it arrives at its destination, the intended recipient, and only the intended recipient, has some way of decrypting it back into the original message ("hello mum").

## Symmetric Encryption: A Story of Passing Secret Notes with a Single Key anchor link

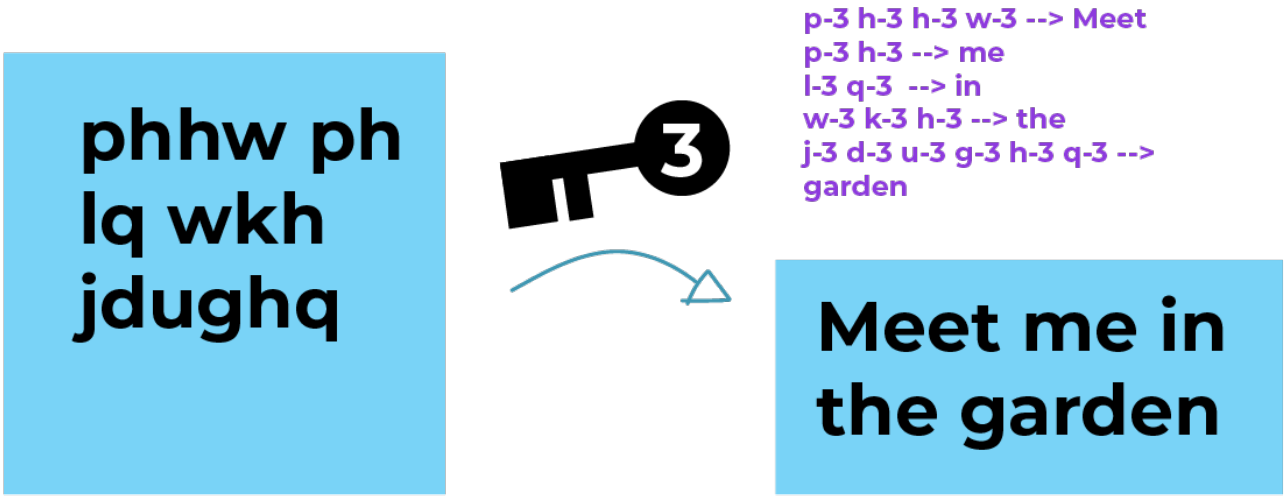
Julia wants to send a note to her friend César that says "Meet me in the garden," but she doesn't want her classmates to see it.



Julia's note passes through a bunch of intermediary classmates before reaching César. Although neutral, the intermediaries are nosy and can easily sneak a peek at the message before passing it on. They are also making copies of this message before passing it on and noting the time at which Julia is sending this message to César.



Julia decides to encrypt her message with a key of 3, shifting the letters down the alphabet by three. So A would be D, B would be E, etc. If Julia and César use a simple key of 3 to encrypt, and a key of 3 to decrypt , then their gibberish encrypted message is easy to crack. Someone could "brute force" the key by trying all the possible combinations. In other words, they can persistently guess until they get the answer to decrypt the message.




The method of shifting the alphabet by three characters is a historic example of encryption used by Julius Caesar: the Caesar cipher. When there is one key to encrypt and decrypt, like in this example where it's a simple number of 3, it is called symmetric cryptography .

The Caesar cipher is a weak form of symmetric cryptography. Thankfully, encryption has come a long way since the Caesar cipher. Using amazing math and the help of computers, a key can be generated that is much, much larger, and is much, much harder to guess. Symmetric cryptography has come a long way and has many practical purposes.

However, symmetric cryptography doesn't address the following issue: what if someone could just eavesdrop and wait for Julia and César to share the key, and steal the key to decrypt their messages? What if they waited for Julia and César to say the secret for decrypting their messages by 3? What if Julia and César were in different parts of the world, and didn't plan on meeting in person?

How can César and Julia get around this problem?

Let's say that Julia and César have learned about public key cryptography. An eavesdropper would be unlikely to catch Julia or César sharing the decryption key—because they don't need to share the decryption key. In public key cryptography, encryption and decryption keys are different.

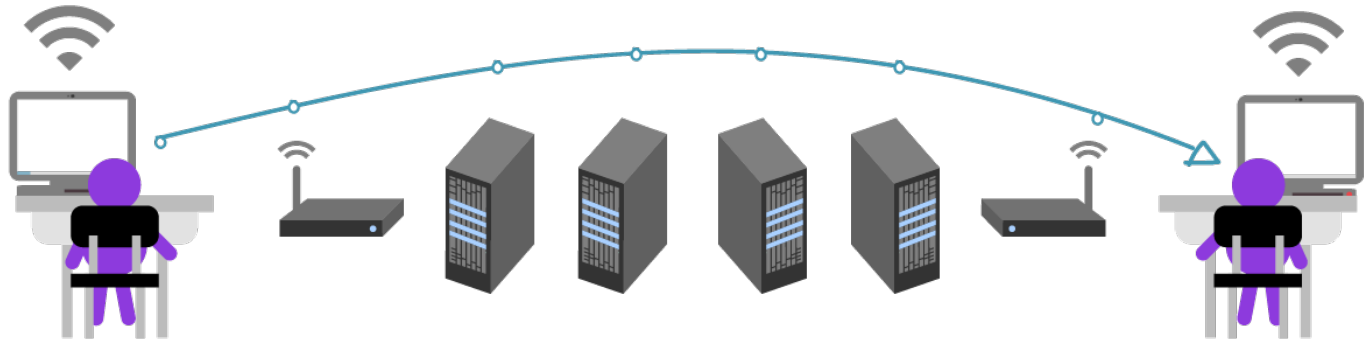
## Public Key Encryption: A Tale of Two Keys anchor link

Let's look at the problem more closely: How does the sender send the symmetric decryption key to the recipient without someone spying on that conversation too? In particular, what if the sender and recipient are physically far away from each other, but want to be able to converse without prying eyes?

Public-key cryptography (also known asymmetric cryptography) has a neat solution for this. It allows each person in a conversation to create two keys—a public key and a private key. The two keys are connected and are actually very large numbers with certain mathematical properties. If you encode a message using a person's public key, they can decode it using their matching private key.

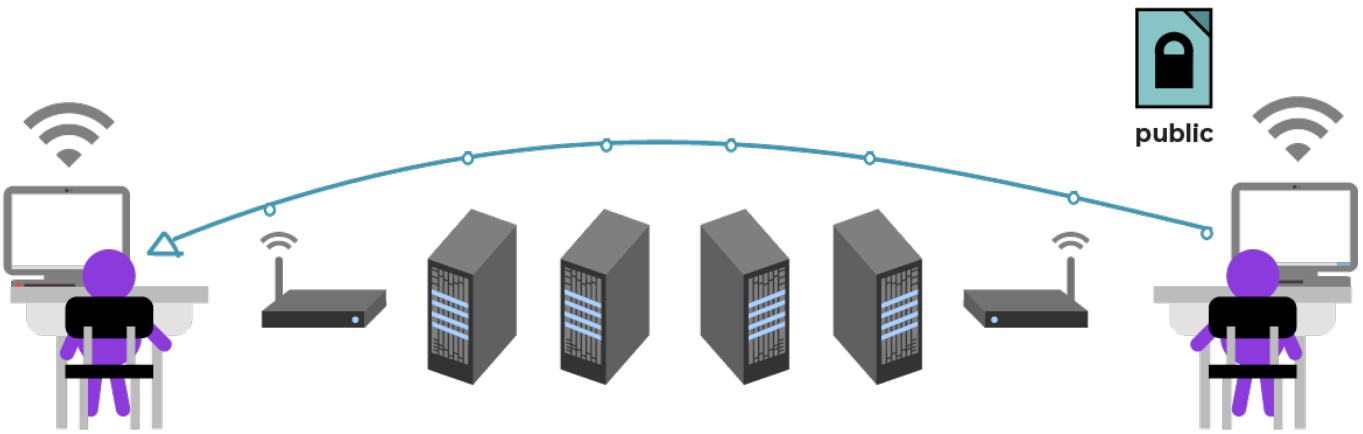
Julia and César are now using their two computers to send encrypted messages using public key cryptography, instead of passing notes. Their classmates passing the notes are now replaced with computers. There are intermediaries between Julia and César: Julia and César's respective Wi-Fi points, Internet Service Providers, and their email servers. In reality, it may be hundreds of computers in between Julia and César that facilitate this conversation. These intermediaries are making and storing

copies of Julia and César's messages each time they are passed through.



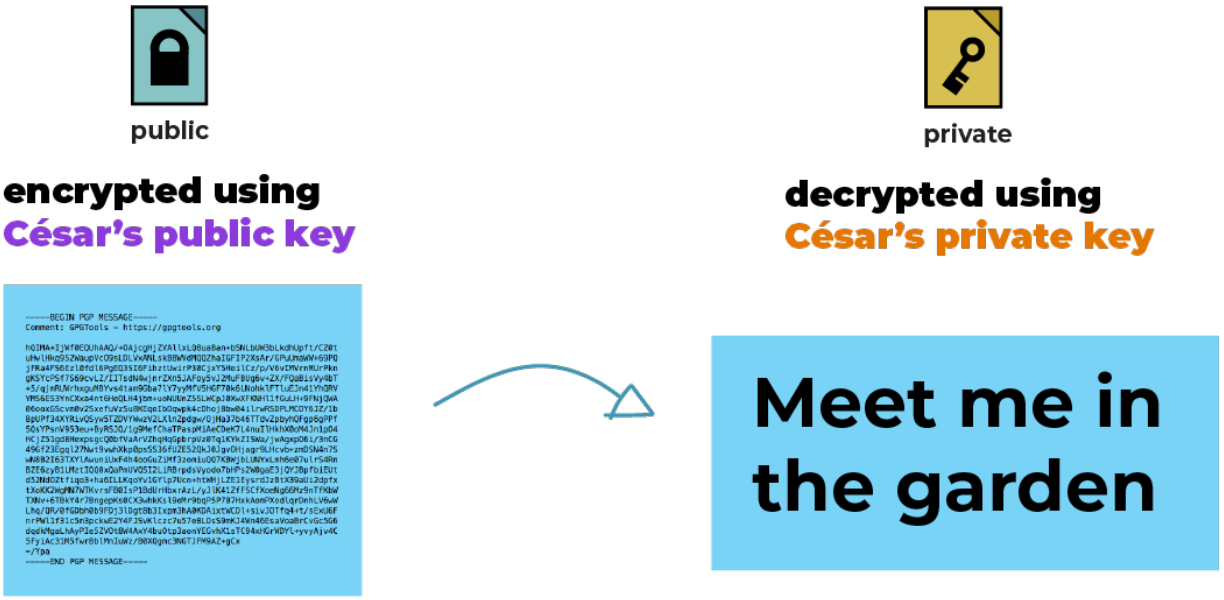
They don't mind that the intermediaries can see them communicating, but they want the contents of their messages to remain private.

First, Julia needs César's public key. César sends his public key (file) over an insecure channel, like unencrypted email. He doesn't mind if the intermediaries get access to it because the public key is something that he can share freely. Note that the key metaphor breaks down around here; it's not quite right to think of the public key as a literal key. César sends the public key over multiple channels, so that the intermediaries can't send one of their own public keys on to Julia instead.

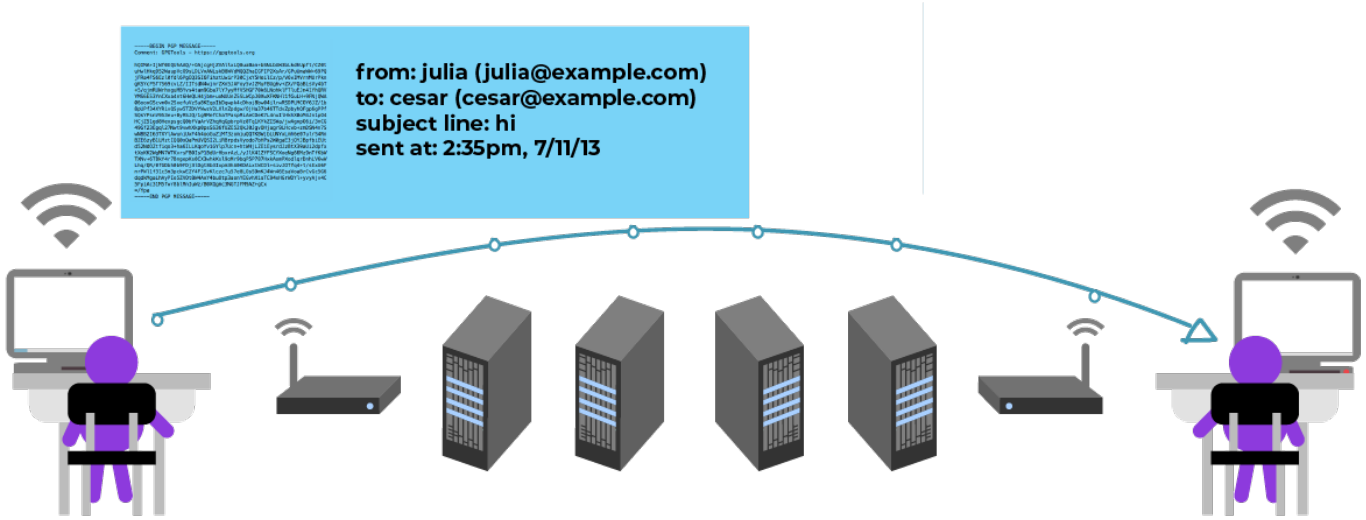



Julia receives César's public key file. Now Julia can encrypt a message to him! She writes her message: "Meet me in the garden."

She sends the encrypted message. It is encrypted only to César.

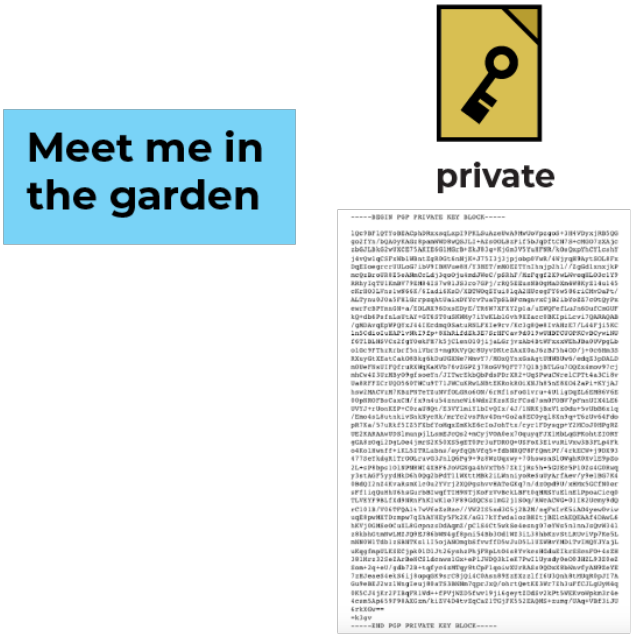



Both Julia and César can understand the message, but it looks like gibberish to anyone else that tries to read it. The intermediaries are able to see metadata , like the subject line, dates, sender, and recipient.




Because the message is encrypted to César's public key, it is only intended for César and the sender (Julia) to read the message.

César can read the message using his private key.




To recap:

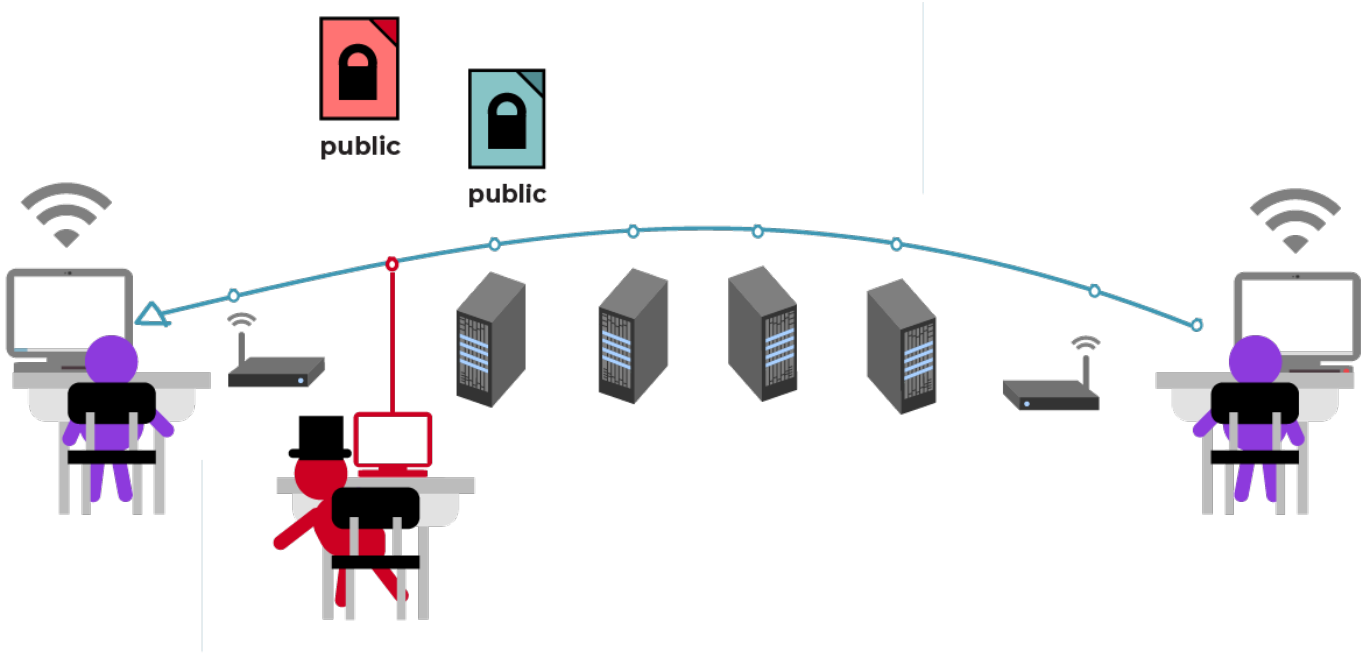
- Public key cryptography allows someone to send their public key in an open, insecure channel.
- Having a friend's public key allows you to encrypt messages to them.
- Your private key is used to decrypt messages encrypted to you.
- Intermediaries—such as the email service providers, Internet service providers, and those on their networks—are able to see metadata this whole time: who is sending what to whom, when, what time it's received, what the subject line is, that the message is encrypted, and so on.

## Another Problem: What About Impersonation? anchor link

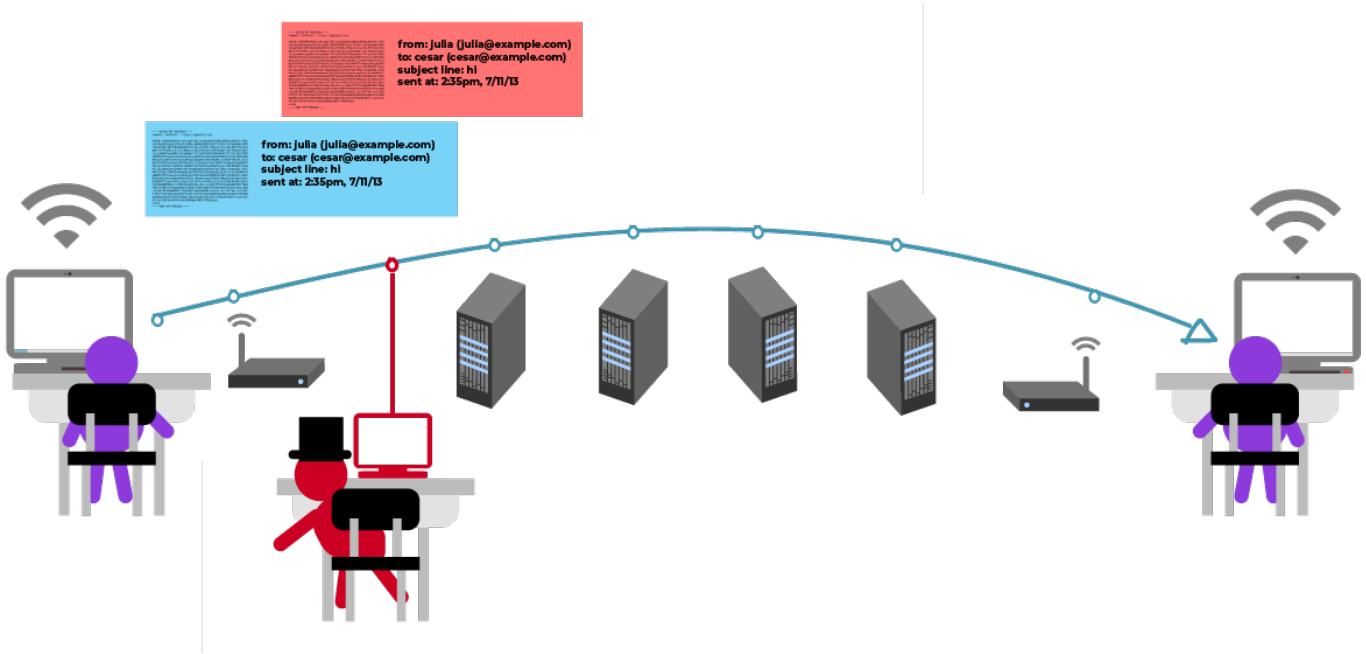
In the example with Julia and César, the intermediaries are able to see metadata this whole time.

Let's say that one of the intermediaries is a bad actor. By bad actor, we mean someone who intends to harm you by trying to steal or interfere with your information. For whatever reason, this bad actor wants to spy on Julia's message to César.

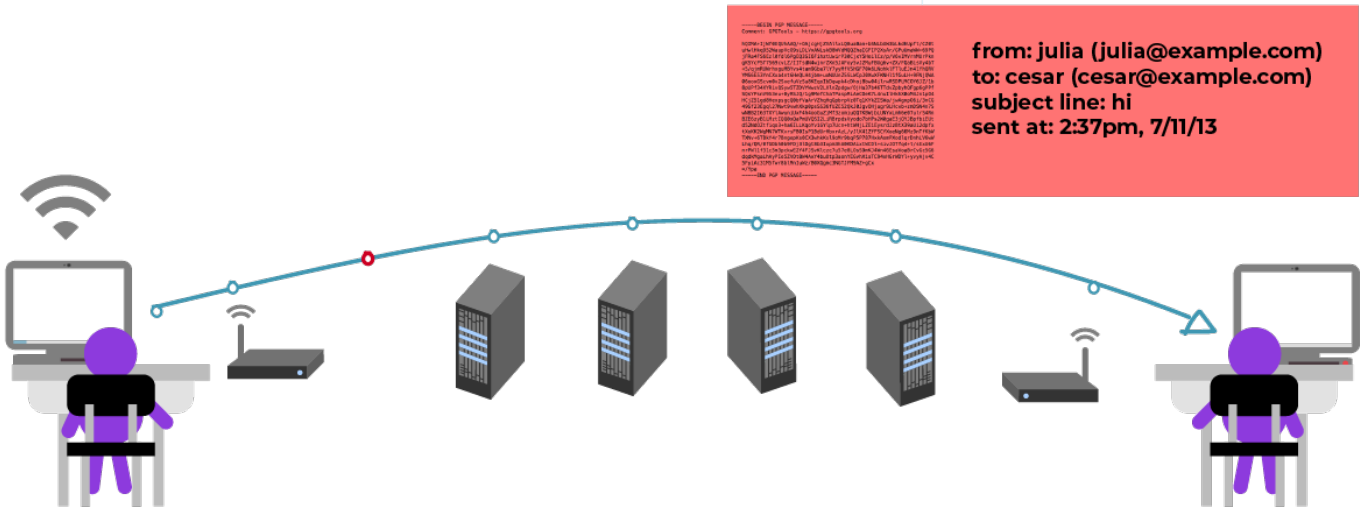
Let's say that this bad actor is able to trick Julia into grabbing the wrong public key file for César. Julia doesn't notice that this isn't actually César's public key. The bad actor receives Julia's message, peeks at it, and passes it along to César.




The bad actor could even decide to change the contents of the file before passing it along to César.




Most of the time, the bad actor decides to leave the contents unmodified. So, the bad actor forwards along Julia's message to César as though nothing has happened, César knows to meet Julia in the garden, and ~gasp~ to their surprise, the bad actor is there too.




This is known as a man-in-the-middle attack . It's also known as a machine-in-the-middle attack .

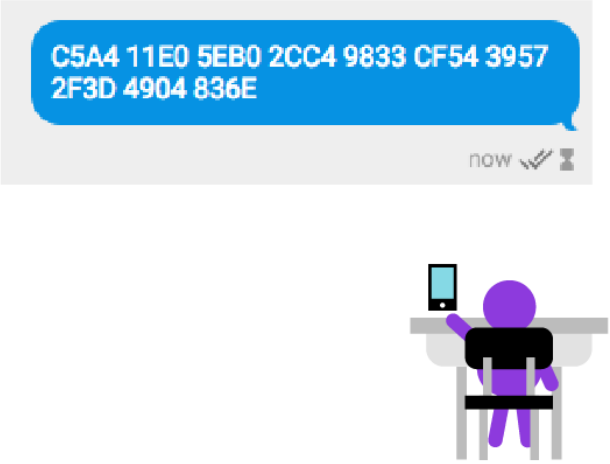
Luckily, public key cryptography has a method for preventing man-in-the-middle attacks.

Public key cryptography lets you double-check someone's digital identity with their real-life identity through something called "fingerprint verification." This is best done in real-life, if you are able to meet with your friend in person. You'd have your public key fingerprint available and your friend double-checks that every single character from your public key fingerprint matches what they have for your public key fingerprint. It's a little tedious, but it's really worth doing.

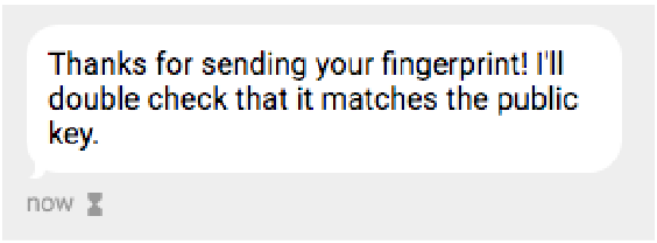
Other end-to-end encrypted apps also have a way to check for fingerprints, though there are some variations on what the practice is called and how it is implemented. In some instances, you'll read each character of the fingerprint extremely carefully and ensure it matches what you see on your screen, versus what your friend sees on their screen. In others, you might scan a QR code on another person's phone in order to "verify" their device." In the example below, Julia and César are able to meet in person to verify their phone fingerprints by scanning each other's QR codes using their phone's camera.

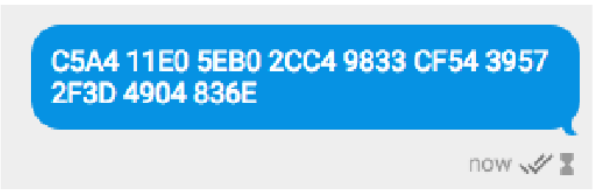
If you don't have the luxury of meeting in person, you can make your fingerprint available through another secure channel, like another end-to-end encrypted messaging app or chat system, or a HTTPS site.

In the below example, César sends his public key fingerprint to Julia using a different end-to-end encrypted app with his smartphone.






To review:

- A man-in-the-middle attack is when someone intercepts your message to someone else. The attacker can alter the message and pass it along or choose to simply eavesdrop.
- Public key cryptography lets you address man-in-the-middle attacks by providing ways to verify the recipient and sender's identities. This is done through fingerprint verification.
- In addition to being used to encrypt a message to your friend, your friend's public key also comes with something called a "public key fingerprint." You can use the fingerprint to verify your friend's identity.
- The private key is used to encrypt messages, as well as for digitally signing messages as you.

## Sign of the Times anchor link

Public key cryptography makes it so you don't need to smuggle the decryption key to the recipient of your secret message because that person already has the decryption key. The decryption key is their private key. Therefore, all you need to send a message is your recipient's matching public, encrypting key. And you can obtain this easily because your recipient can share their public key with anyone, since public keys are only used to encrypt messages, not decrypt them.

But there's more! We know that if you encrypt a message with a certain public key, it can only be decrypted by the matching private key. But the opposite is also true. If you encrypt a message with a certain private key, it can only be decrypted by its matching public key.

Why would this be useful? At first glance, there doesn't seem to be any advantage to sending a secret message with your private key that everyone who has your public key can decrypt. But suppose you wrote a message that said "I promise to pay Aazul $100," and then turned it into a secret message using your private key. Anyone could decrypt that message—but only one person could have written it: the person who has your private key. And if you've done a good job keeping your private key safe, that means you, and only you, could've written it. In effect, by encrypting the message with your private key, you've made sure that it could have only come from you. In other words, you've done the same thing with this digital message as we do when we sign a message in the real world.

Signing also makes messages tamper-proof. If someone tried to change your message from "I promise to pay Aazul $100" to "I promise to pay Ming $100," they would not be able to re-sign it using your private key. So, a signed message guarantees it originated from a certain source and was not messed with in transit.

## In Review: Using Public Key Cryptography anchor link

Let's review. Public key cryptography lets you encrypt and send messages safely to anyone whose public key you know.

If others know your public key:

- They can send you secret messages that only you can decode using your matching private key and,
- You can sign your messages with your private key so that the recipients know the messages could only have come from you.

And if you know someone else's public key:

- You can decode a message signed by them and know that it only came from them.

It should be clear by now that public key cryptography becomes more useful when more people know your public key. The public key is shareable, in that it's a file that you can treat like an address in a phone book: it's public, people know to find you there, you can share it widely, and people know to encrypt messages to you there. You can share your public key with anyone who wants to communicate with you; it doesn't matter who sees it.

The public key comes paired with a file called a private key. You can think of the private key like an actual key that you have to protect and keep safe. Your private key is used to encrypt and decrypt messages.

It should also be apparent that you need to keep your private key very safe. If your private key is accidentally deleted from your device, you won't be able to decrypt your encrypted messages. If someone copies your private key (whether by physical access to your computer, malware on your device, or if you accidentally post or share your private key), then others can read your encrypted messages.

They can pretend to be you and sign messages claiming that they were written by you.

It's not unheard of for governments to steal private keys off of particular people's computers (by taking the computers away, or by putting malware on them using physical access or phishing attacks). This undoes the protection private key cryptography offers. This is comparable to saying that you might have an unpickable lock on your door, but somebody might still be able to pickpocket you in the street for your key, copy the key and sneak it back into your pocket and hence be able to get into your house without even picking the lock.

This goes back to threat modeling : determine what your risks are and address them appropriately. If you feel that someone would go through great trouble to try to get your private key, you may not want to use an in-browser solution to end-to-end encryption. You instead may opt to just have your private key stored on your own computer or phone, rather than someone else's computer (like in the cloud or on a server).

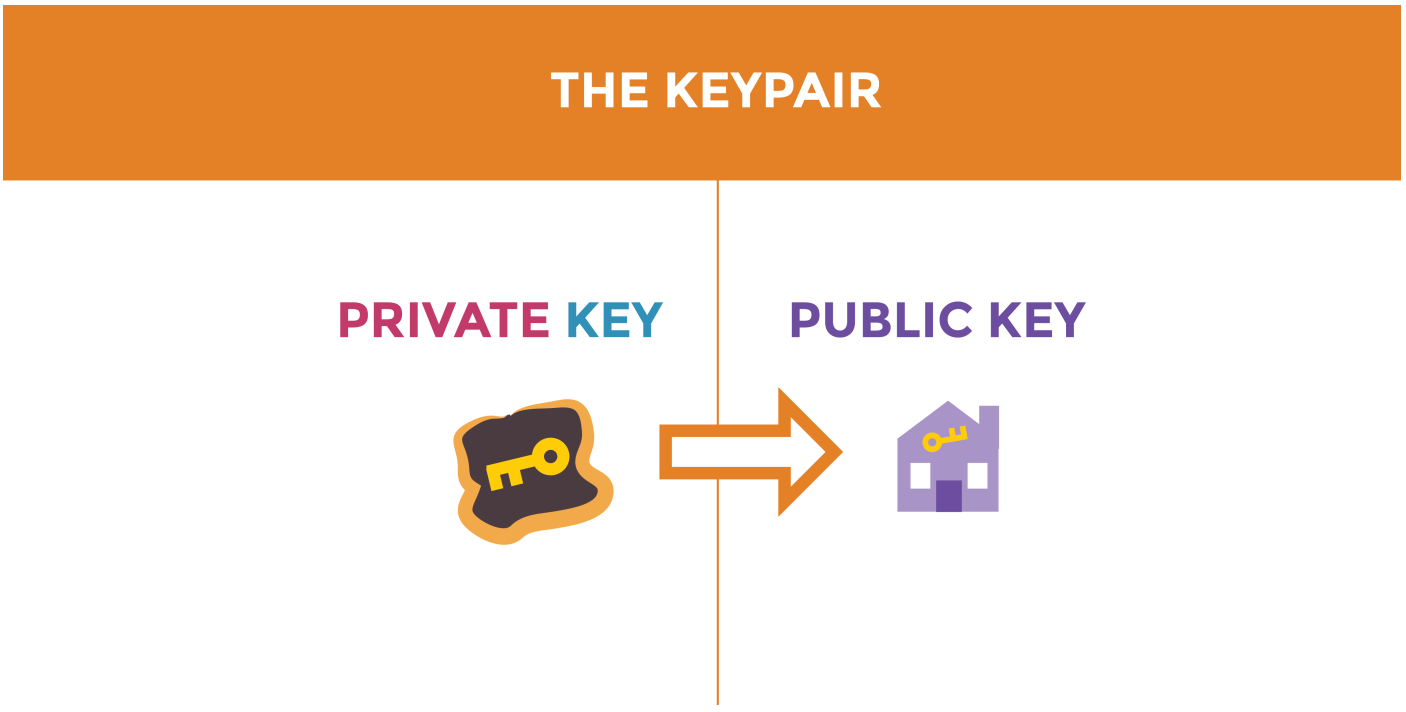
## Review of Public Key Cryptography, and A Specific Example: PGP. anchor link

So, we went over symmetric encryption and public key encryption as separate explanations. However, we should note that public key encryption uses symmetric encryption as well! Public key encryption actually just encrypts a symmetric key, which is then used to decrypt the actual message.

PGP is an example of a protocol that uses both symmetric cryptography and public key cryptography (asymmetric). Functionally, using end-to-end encryption tools like PGP will make you very aware of public key cryptography practices.

## What Exactly Are Keys. And How Are Keys Tied Together? anchor link

Public key cryptography is based on the premise that there are two keys: one key for encrypting, and one key for decrypting. How it basically works is you can send a key over an insecure channel, like the Internet. This key is called the public key. You can post this public key everywhere, in very public places, and not compromise the security of your encrypted messages.

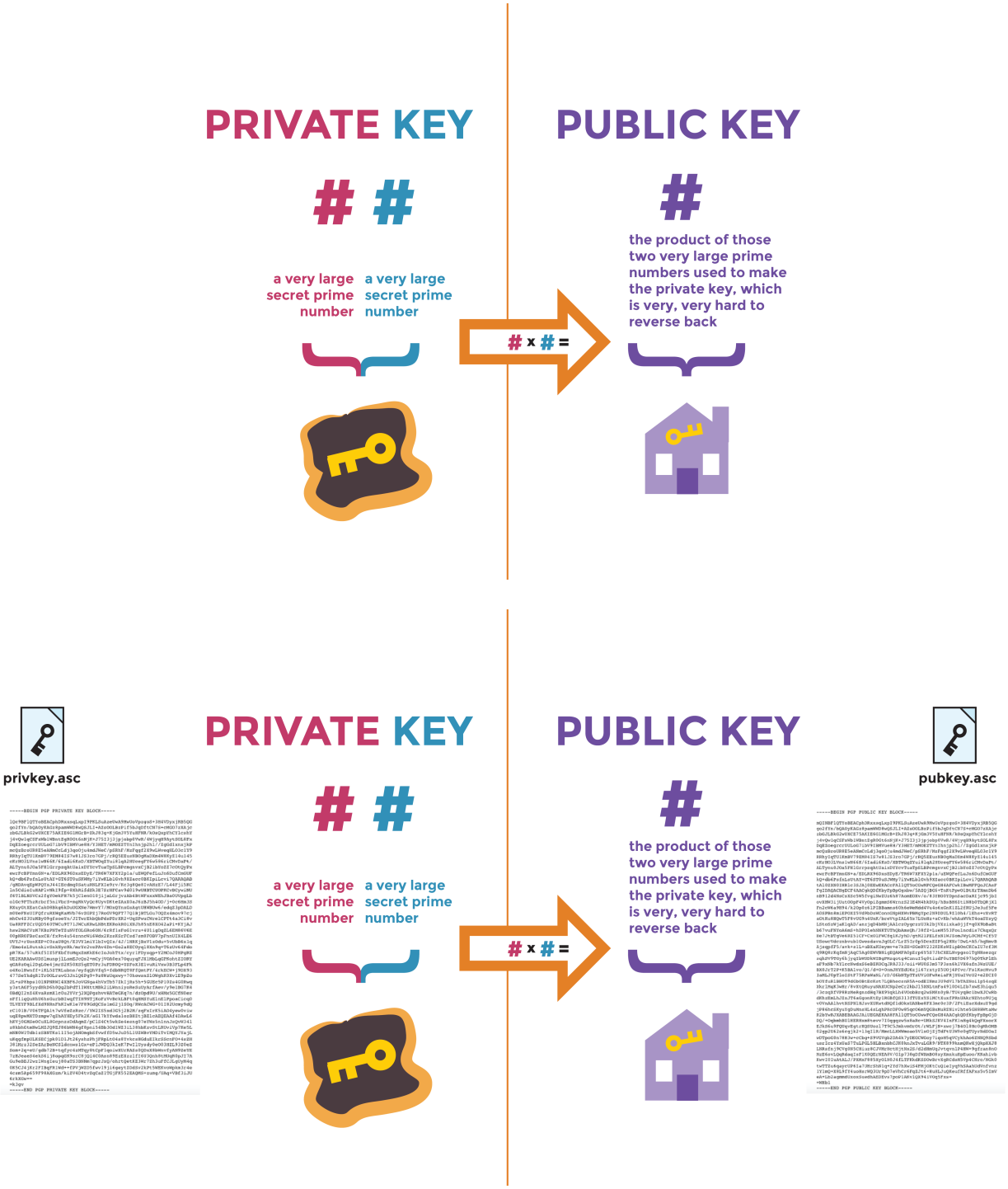



This shareable key is the public key: a file that you can treat like an address in a phone book: it's public,

people know to find you there, you can share it widely, and people know to encrypt to you there.

The public key comes paired with a file called a private key. You can think of the private key like an actual key that you have to protect and keep safe. Your private key is used to encrypt and decrypt messages.

We're going to examine the key generation in a commonly-used public key cryptography algorithm called RSA (Rivest–Shamir–Adleman). RSA is often used to generate key pairs for PGP encrypted email.




The public key and private key are generated together and tied together. Both rely on the same very large secret prime numbers. The private key is the representation of two very large secret prime numbers. Metaphorically, the public key is the product number: it is made up of the same two very large prime numbers used to make the private key. What's amazing is that it's very hard to figure out which two large prime numbers created the public key.

This problem is known as prime factoring, and some implementations of public key cryptography take

advantage of this difficulty for computers to solve what the component prime numbers are. Modern cryptography allows us to use randomly chosen, ridiculously gigantic prime numbers that are hard to guess for both humans and computers.

And, the strength here is that people can share their public keys over insecure channels to let them encrypt to each other! In the process, they never reveal what their private key (secret prime numbers) is, because they never have to send their private key for decrypting messages in the first place.

Remember: For public key cryptography to work, the sender and the recipient need each other's public keys.

Another way you can think of it: The public key and private key are generated together, like a yin-yang symbol. They are intertwined.


The public key is searchable and shareable. You can distribute it to whoever. You can post it on your social media, if you don't mind that it reveals the existence of your email address. You can put it on your personal website. You can give it out.

The private key needs to be kept safe and close. You just have one. You don't want to lose it, or share it, or make copies of it that can float around, since it makes it harder to keep your private messages private.

## How PGP Works anchor link

Let's see how public key cryptography might work, still using the example of PGP. Let's say you want to send a secret message to Aarav:

1. Aarav has a private key and, like a good public key encryption user, he has put its connected public key on his (HTTPS) web page.
2. You download his public key.
3. You encrypt your secret message using Aarav's public key and send it to him.
4. Only Aarav can decode your secret message because he's the only one with the corresponding private key.

Pretty Good Privacy is mostly concerned with the minutiae of creating and using public and private keys. You can create a public/private key pair with it, protect the private key with a password , and use it and your public key to sign and encrypt text.

If there's one thing you need to take away from this overview, it's this: Keep your private key stored somewhere safe and protect it with a long passphrase .

## Metadata: What Public Key Encryption Can't Do anchor link

Public key encryption is all about making sure the contents of a message are secret, genuine, and untampered with. But that's not the only privacy concern you might have. As we've noted, information about your messages can be as revealing as their contents (See “metadata”).

If you exchange encrypted messages with a known dissident in your country, you may be in danger for simply communicating with them, even if those messages aren’t decoded. In some countries you can face imprisonment simply for refusing to decode encrypted messages.

Disguising that you are communicating with a particular person is more difficult. In the example of PGP, one way to do this is for both of you to use anonymous email accounts, and access them using Tor. If you do this, PGP will still be useful, both for keeping your email messages private from others, and proving to each other that the messages have not been tampered with.

Now that you’ve learned about public key cryptography, try out using an end-to-end encryption tool like Signal for iOS or Android.