What is plausible deniability (in encryption) and does it really work?

Josh Lake ⋮ 40-51 minutes ⋮ 10/27/2020

Specialist in Security, Privacy and Encryption

Updated: January 8, 2024

Plausible deniability seems like the ultimate get-out-of-jail-free card. But does it really work when it comes to encryption?

Encryption and sound password practices may be enough to protect your data in most scenarios, but are there situations where you may need plausible deniability as well?

Strong encryption can keep your adversaries out, but only if they can't obtain your password. If your threat model includes coercion, either by the threat of jail time from the government, or through torture, then you may give in and hand over the key to save yourself. This would leave the data in the hands of your adversary, potentially exposing you to future consequences.

But what if you could protect yourself from coercion without having to hand over your key? What if you could deny the existence of the data in the first place? What if there was no clear evidence of the data?

It can be much harder to compel you to hand over a key to something if your adversary isn't sure whether or not it even exists.

This is the concept behind plausibly deniable encryption. While it may be helpful in certain situations, it isn't the straightforward get-out-of-jail-free card that some think.

The concept of plausible deniability

The concept of plausible deniability is rooted in politics and espionage, and involves a person having the ability to deny their involvement, responsibility or knowledge of certain events or information.

It can involve conducting actions in a way that leaves no evidence, or specifically altering systems around certain individuals to allow them to truthfully deny their awareness of what took place. It can also involve destroying evidence, so that a particular action is plausibly deniable.

A good example of plausible deniability would be someone asking a journalist to go off-the-record so that they could anonymously leak information. The leaker would have plausible deniability as long as:

- The leaked information was known by multiple people.
- The journalist had turned off the recorder and there was no evidence.
- No one saw the two people together.
- The journalist did not betray the source.

Another example would be if a President purposely left a room before a sinister plan is discussed. The President could then honestly say that they had no knowledge of the plan, protecting them from incrimination.

In these scenarios, people may have suspicions and be near-certain of who is responsible, or that a scheme had tacit approval. There may only be one reasonable explanation. But as long as there is no evidence and a smidgen of doubt, then action can't be taken against the parties in many circumstances.

Plausible deniability can protect figures from prosecution or even public outrage. Whether it is effective or not really depends on the specific situation. The concept is similar when it is applied in cybersecurity.

Plausibly deniable encryption in cybersecurity

In the realm of cybersecurity, plausibly deniable systems aim to give their users the ability to deny the existence of data. This is known as deniable encryption, the focus of our article, but there is another related concept known as deniable authentication. This casts doubt over whether or not messages are authentic.

Deniable encryption can be used to hide data within file-hosted containers, partitions, and hard drives. It combines both steganography and cryptography to grant its users plausible deniability over whether or not the data exists. It has the potential to keep data safe from adversaries, and in certain situations it could protect people from imprisonment or violence.

However, in many scenarios, the effectiveness of deniable encryption is debated.

Rubberhose

Rubberhose was the first freely available deniable encryption program. It was released in 1997 by a pre-WikiLeaks Julian Assange, alongside researchers Suelette Dreyfus and Ralph Weinmann. The program was named, with a hint of dark humor, after the practice of continually beating someone with a rubber hose. Rubber-hose cryptanalysis refers to someone being coerced by torture into handing over encryption keys.

According to the software documentation, Rubberhose was designed with the intention of protecting human-rights workers and activists from torture, particularly in repressive regimes.

The program made it possible to encrypt drives while obscuring how much actual data was encrypted. This gave users plausible deniability for the encrypted material.

Rubberhose accomplished this by first writing random characters to the entire drive. To any observer, these characters look exactly the same as encrypted data. Users could set up separate partitions, each with different passwords. The encrypted data for each partition was broken up by Rubberhose and scattered.

Due to this structure and the random noise that was written to the entire disk, neither physical disk testing or mathematical analysis revealed how many encrypted partitions were on the drive.

In theory, this allowed the user to hide their data from adversaries. However, it suffered from many of the same complications that we will discuss at the end of this article when we cover various threat scenarios.

Other deniable encryption programs

Rubberhose popularized the concept of deniable encryption, and a range of other programs in the same vein popped up around the same time that Rubberhose fell out of regular maintenance. The most prominent ones included:

- **Phonebook File System** – This disk encryption software offered plausible deniability and was strongly inspired by Rubberhose. However, it is no longer in active development.
- ***FreeOTFE*** – Developed by Sarah Dean, FreeOTFE (on-the-fly disk encryption) is an open-source disk encryption program that was launched in 2004. It offered deniable encryption, however no updates to the project have been made since 2010, so it shouldn't be used.
- **LibreCrypt** – LibreCrypt began as a fork of FreeOTFE after its development stalled. At this stage, LibreCrypt also seems to have been abandoned, so alternatives should be used instead.
- **BestCrypt** – BestCrypt is encryption software developed by Jetico, a company that has been in the business since 1995. It's based in Finland, trusted by major organizations and publishes its source code, so that researchers can scrutinize it. If you are looking for an option with good support and don't mind paying a little, BestCrypt is a good choice. A free trial lasting 21 days is available for those who only need a short-term solution.
- **TrueCrypt** – This encryption package was launched in 2004, and despite some initial licensing issues, it became immensely popular. It was free, its source code was available for scrutiny and it offered plausible deniability. However, in 2014 it was abruptly shut down, notifying its users that it was no longer safe. While some security problems were eventually found, the sudden halt to such a widely used program generated a range of rumors. A number of successors were created in its wake, the most notable of which is VeraCrypt.
- **DiskCryptor** – DiskCryptor was originally developed by a former TrueCrypt user, but development stalled in 2014. The project has since been picked up by another developer using a pseudonym, but it is not widely used or scrutinized. Because not much is known about the developer and it isn't widely tested, it is probably best to use other options.
- **CipherShed** – As a fork of TrueCrypt, CipherShed was open source and offered deniable encryption. However, the project has since been abandoned, so it should not be used.
- **Fuyoal** – This encryption package offers plausible deniability and is still actively being developed. However, it isn't widely used and hasn't undergone heavy scrutiny, so it's probably best to stick to alternatives.

Most of the options we just mentioned have either been discontinued, or are still such small projects that they haven't been put to the test by the wider community. The only one that we can recommend for plausible deniability is the paid option, BestCrypt. We have simply included the rest to give you a bit of background, as well as a quick understanding of which tools not to use.

This leaves us to discuss the only free major player, and the most commonly used option, VeraCrypt.

See also: Best disk encryption software

VeraCrypt

As we mentioned, VeraCrypt is a fork of TrueCrypt that is freely available for Windows, MacOS and Linux. It was launched in 2013 and allows its users to encrypt their entire drives; create hidden containers, hidden operating systems and more.

It includes patches to vulnerabilities that were found in TrueCrypt, and was last audited in 2016. It's popular in the security community and the only widely scrutinized free disk encryption package that offers plausible deniability.

Many security nerds even recommend it for disk encryption over more common options like Apple's FileVault or Microsoft's Bitlocker. This is partly because big tech companies don't open up the bulk of their code to the community, and some are suspicious that this could make it easier to hide government backdoors.

While it may seem a little less slick, VeraCrypt is well-regarded for its security, especially among the overly paranoid. But let's not get too bogged down in its general features and focus on its plausible deniability instead.

VeraCrypt's plausibly deniable encryption

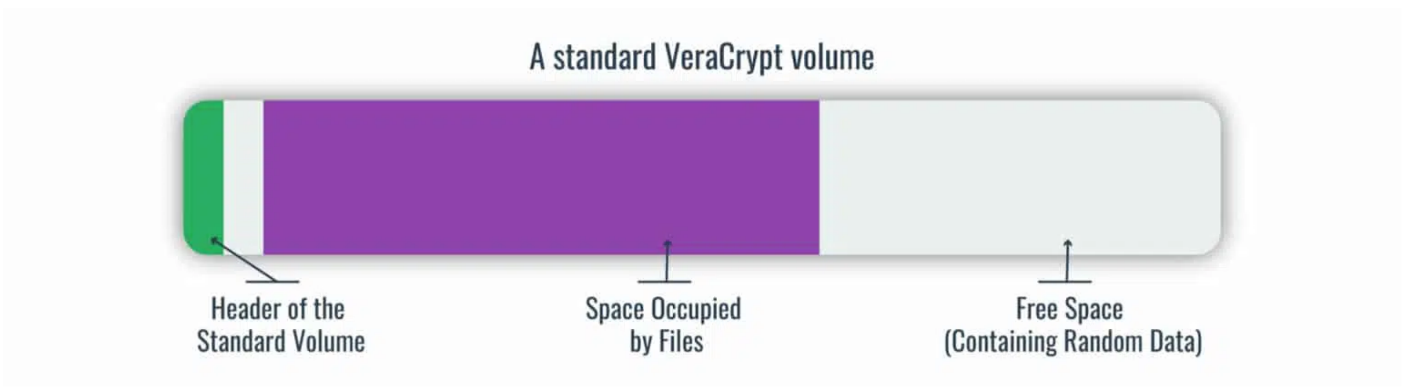
VeraCrypt offers two different types of encrypted volumes (storage areas) for protecting your files:

- **File-hosted encrypted volumes** (also known as containers) – These act like normal files, and they can be kept on any type of storage device. Each one contains an independent encrypted virtual disk device.
- **Partition or full disk encryption volumes** [referred to as device-hosted (non-system) in the documentation] – A partition is a section of a hard drive that has been divided up by its user. The operating system essentially treats each partition as though they are separate hard drives, even though they are both on the same physical disk. Device-hosted volumes encrypt the entire hard disk, flash drive or other storage device.

Both of these types of standard volume can store hidden volumes inside them. Hidden volumes are how VeraCrypt achieves plausible deniability. But first, let's explain how standard volumes work.

Standard Volumes

You can easily create VeraCrypt volumes to encrypt your files by downloading the software and following the wizard's instructions. A standard VeraCrypt volume is structured like this:


A standard VeraCrypt volume
Header of the Standard Volume
Space Occupied by Files
Free Space (Containing Random Data)

- The header contains important information like the size, the key and other specifics.
- To the left is a sliver of random data, which will be explained when we go into the details of hidden volumes.
- Next comes the space occupied by your files, which varies according to how much data you have

on it.
- Finally comes the free space. This is either the rest of the space on your partition or hard drive, or the remainder of the container size if it’s a file-hosted volume.

When a VeraCrypt volume is created, the entirety of the volume is encrypted, from the header, to the files, to the free space. This means that the entire thing looks like random data, and someone looking from the outside without the password cannot see that there is a header, files and free space contained in the muddle of data, nor can they see how much is used and how much is open.

The volume itself contains no sign or signature that it’s a VeraCrypt volume, so an outsider without the password has no way of telling what is in there or how much of it. This is an important aspect of VeraCrypt’s plausible deniability.

However, one major complication is that a whole bunch of random data looks strange to any investigator that knows what they are doing. For the most part, people don’t just have piles of random data on their hard drives, so any forensic examiner will probably assume that it’s encrypted data. But they won’t have solid proof.

We’ll cover the details of how effective the deception is further down in the article. For now, the important thing is that you understand what a standard VeraCrypt volume is, and the basics of how it looks from the outside.

VeraCrypt hidden volumes

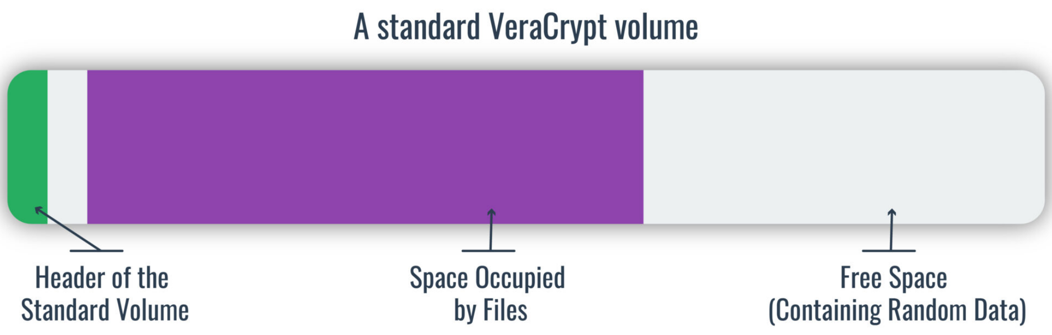
As we mentioned, VeraCrypt hidden volumes are what can give you plausible deniability for your data. If your adversary takes your device and asks for your key, you can simply offer them the key to the standard volume that houses the hidden volume inside it.

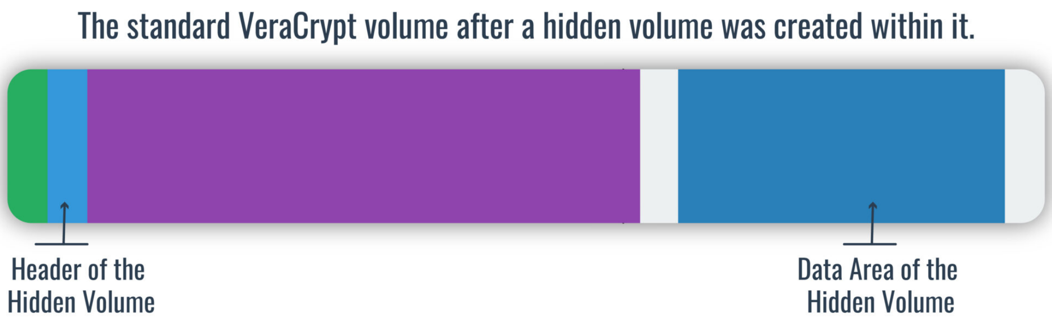
The idea behind it is that you would pack the standard volume with a bunch of files that may look sensitive, but that you don’t actually care about if they fall into your adversary’s hands. You hope that when you give them the key, they have a look around, see that there is nothing too valuable or incriminating there, and then send you on your way.

Meanwhile, your sensitive data is lurking just below in a hidden volume that they have no idea about.

How do hidden volumes work?

Remember our picture of a standard VeraCrypt encrypted volume from before? Well, here it is again, alongside a diagram of a standard volume that contains a hidden volume within:


A standard VeraCrypt volume
Header of the Standard Volume
Space Occupied by Files
Free Space (Containing Random Data)


The standard VeraCrypt volume after a hidden volume was created within it.
Header of the Hidden Volume
Data Area of the Hidden Volume

As you can see, the header of the hidden volume is slotted in right next to the header of the standard volume, in the space that is otherwise kept free. The data for the hidden volume lies to the right of the files occupied by the standard volume. It takes up the space that was previously free and contained random data.

Note that these diagrams aren't drawn to any scale. While the headers are always 512 bytes each, the space occupied by files will vary depending on how much data you store on the volume. The free space will be the remainder of your hard drive, partition, or whatever the size of the container you set was. The hidden volume will take up however much space you set aside for it, including both its data and its free space.

Just like before, to an outsider without the password, all of this data just looks completely random, including the hidden volume. There is no sign that it has been encrypted by VeraCrypt.

The central idea behind hidden volumes is that when the outer volume is mounted, it's impossible to prove whether or not a hidden volume is inside it (as long as the right precautions are taken). While this can certainly be true, we will discuss just how useful it can be in practical scenarios later in the article.

It's impossible to prove, because as we mentioned earlier, VeraCrypt always fills up the entire volume (whether it's file-hosted, a partition or an entire drive) with encrypted data, even the free space. To anyone without the password, there is no way to distinguish the free space from the actual files.

The system works because when VeraCrypt mounts a volume, it starts by reading the first header, the one for the standard volume. It moves on to reading the space where a second header would be, *if there is a hidden volume*. It then tries to decrypt the first header's data with the password.

If the user entered the correct password for the header of the standard volume, the data is decrypted, and the program knows it has been successful because certain data lines up. It then retrieves the key and other specifications from the header, and uses them to decrypt the rest of the standard volume. This grants the user access to the standard volume.

If the password was unsuccessful, VeraCrypt attempts to follow the same process with the data from where a hidden volume would be. If there is a hidden volume and the right password has been entered, the hidden volume's header will be decrypted, and VeraCrypt can see that the right values match up.

The program then takes the key and other specifications from the hidden volume's header, and uses them to decrypt the hidden volume. The user can then access the data in the hidden volume.

If the password doesn't lead to the header data matching up for either the standard volume or the hidden volume, then VeraCrypt assumes that the wrong password has been entered and terminates the mounting process. The user will have to try typing in their password again.

This system, as well as the practice of encrypting the headers of both standard volumes and hidden volumes, means that people cannot obtain any revealing information unless they have the password to the standard volume.

It also means that handing over the password for the standard volume doesn't give anything away to the adversary – VeraCrypt follows the same mounting and decryption process either way.

Hidden operating systems

Software like VeraCrypt can also create entire hidden operating systems, potentially granting plausible deniability to a user's entire OS. However, it requires an unencrypted copy of the VeraCrypt boot-loader on either the system drive or a system disk.

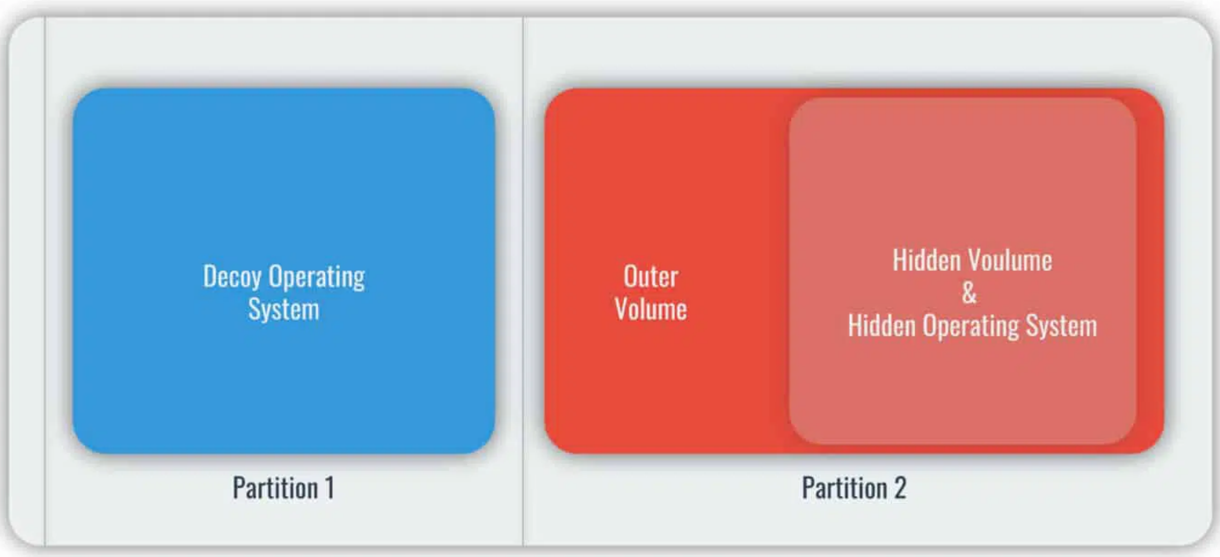
Obviously, if you have the boot-loader on one of your devices and it falls into an adversary's hands, they are going to become suspicious and want to know what you are hiding.

To throw them off, you need to set up a decoy operating system through the VeraCrypt wizard. Ideally, you will regularly use this decoy operating system for mundane activities, saving the hidden OS for when you are doing anything sensitive.

Otherwise, your adversary may become suspicious about why you have an operating system that is barely being used. This could lead to them suspecting a hidden OS and torturing you or hanging severe legal penalties over your head.

If the decoy operating system seems to show relatively normal computer activity, it may just be enough to fool your adversary that you don't have whatever it is they are looking for hidden on your device.

To complete the deception, the hidden operating system is placed within a standard VeraCrypt volume. This outer volume acts as a normal VeraCrypt volume with a third password. This allows you to hide the hidden OS from your adversary. Remember, VeraCrypt encrypts entire volumes – even the free space – with random data, so they can't see what's on there.



How a hidden operating system is structured.

Keep some semi-sensitive files that you don't mind your adversary getting their hands on in this outer volume. This allows you to hand over the third password in the hopes that your adversary will assume that the decoy system and the encrypted volume are all that you have, and not suspect the hidden OS that's lurking just out of reach.

How VeraCrypt's hidden operating systems work

Just like with hidden volumes, you will have multiple passwords, one for the decoy operating system, another for the hidden OS and a third for the outer volume. The process for booting either your decoy or your hidden operating system works similarly to how VeraCrypt mounts either standard or hidden volumes, depending on the password that you enter.

When a password is entered into VeraCrypt's boot loader, it attempts to decrypt the first 512 bytes of the first logical track of the system drive. This is where it stores the master key for drives and partitions that aren't hidden. If it's successful, it boots the decoy OS.

If this is unsuccessful and there is a partition behind the active one, VeraCrypt attempts to use the same password to decrypt the location where a hidden volume's header would be stored (if one was present). If the header is decrypted, it uses its information to mount the hidden volume, allowing the user to access their hidden operating system instead.

Precautions for safeguarding your data & plausible deniability

No amount of technology can protect your data and give it plausible deniability if the right precautions aren't taken. There's basic stuff, like physical security – things like making sure you don't leave your laptop while you're logged in to your hidden operating system or have your hidden volume decrypted.

You also need to be wary of unencrypted data being stored in your RAM until the device is turned off. Malware is another big issue. If you don't follow security best practices and end up with malware on your computer, then VeraCrypt may not be able to keep your data safe.

VeraCrypt has a long list of precautions to take in its official documentation, but this article will get very tedious if we list them all out. Please familiarize yourself with the Security Requirements and Precautions page if you are an activist, journalist or anyone else who needs to keep their data safe from sophisticated adversaries. This page on security requirements for hidden volumes is also crucial.

Prepare your excuses

If your device ends up in the hands of a technically literate adversary, you need to be prepared to explain why your drive has seemingly random data, or why you have the VeraCrypt boot loader:

- If you are questioned about a drive or a partition filled with random data, you can explain that it's because you wiped the previous data with VeraCrypt or a similar program.
- If you have two VeraCrypt-encrypted partitions on a system drive, and your adversary becomes suspicious, you can say that you wanted to separate your system files from your non-system files. This is often done because if one partition gets damaged, the other will still be protected. Alternatively, you could claim that this setup makes it easier to reinstall the system without losing documents.
- The same problem can also be addressed by claiming that one partition is encrypted with a highly secure cascade algorithm, while the other is non-cascading. This allows you to have the highest level of security for incredibly important files, while the other partition can be encrypted with a much faster algorithm that can still offer a reasonable degree of security to the files. A similar tactic involves saying that the partition with the most sensitive documents has an incredibly long, yet tiresome password for security, while the other partition is protected by a shorter and easier password.

Does plausible deniability really work?

It really depends on the kinds of scenarios you may potentially face, and there is a lot of complexity. Although the bulk of this article has focused on VeraCrypt, it's only because it's the most common and thoroughly vetted encryption system that offers the possibility of plausible deniability. The following information should apply to all current types of software that offer deniable encryption (that are used widely enough to have been audited, and presumed safe by the security community).

Scenario 1: Your adversary is your country's authorities and you live in a democracy without key disclosure laws

In recent years, many countries have introduced key disclosure laws, which compel suspects to hand over their passwords under the threat of heavy legal penalties. However, some democracies such as Switzerland, Finland and Belgium have not introduced these laws.

If you live in one of these countries, the authorities cannot force you to hand over your password, even if they suspect you of heinous crimes such as terrorism. In these cases, plausible deniability is unnecessary. All you have to do is encrypt the data with a strong password and follow security best practices to keep it safe.

If they ask you to hand over the key, you can simply say no. Even if the authorities are certain that you have the data they need on your device, they cannot compel you to give it to them.

This means that there isn't much of a reason to go to the effort of setting up a plausibly deniable encryption system if you are a resident of these countries, and your threat model's adversaries only include the authorities.

Scenario 2: Your adversary is your country's authorities and you live in a democracy with key disclosure laws

Countries such as Australia, the United Kingdom and the United States of America have introduced laws that force suspects to hand over their encryption keys. If they refuse, they face lengthy jail sentences.

In these jurisdictions, plausibly deniable encryption sounds like a good way to protect yourself. After all, if the authorities can't prove whether or not data exists, they can't send you to jail for not handing over the keys to it, can they?

Throughout our research, we could find no examples of plausibly deniable encryption being tested by the courts in any jurisdiction. This means that we cannot know for sure how it would play out if you had a plausibly deniable setup.

However, we can still speculate based on the evidence left behind. Bear in mind that the following discussion only applies if you have used VeraCrypt properly and taken all of the necessary precautions. If not, the software probably can't help you.

If you commit a serious enough crime and the authorities believe that you have evidence on your devices, they will send them to a forensic expert. What they will see depends on how you have used VeraCrypt to achieve plausible deniability:

A hidden volume in a file-hosted container

Experts have software that can sort for files that look like they are encrypted. They will find the outer volume of the container that has the hidden volume within. They will see the random data, which is pretty unusual, and then note that it has a size divisible by 512. Anyone working as an examiner will know that the size of VeraCrypt files is always in multiples of 512.

At this stage, they will be pretty confident that you have encrypted data. They may not be 100 percent certain, but it's probably beyond reasonable doubt. Even VeraCrypt admits that there is "…practically no plausible explanation for the existence of a file containing solely random data." There is a chance that you could be thrown in jail for contempt if you still refused to hand over the password once the forensics examiner had found the encrypted file container.

If you conceded and handed over the password, the examiner could go into the encrypted volume and check it out. If you had some dummy files in there to throw them off, they may just be enough for them to conclude that there is nothing else within the volume, or at least nothing that they could prosecute you over, because there is no proof of the hidden volume.

VeraCrypt always encrypts the free space in a container, so there are no traces of it. They may be suspicious if you have a ton of empty space in the container, but you could always argue that you made it that large because you expected to accumulate more files over time.

If you didn't put any fake sensitive files in the container to deceive them, then they may not believe your excuses. After all, why would anyone create an encrypted container for nothing? If this is the case, your plausible deniability is on shaky ground, and you could end up in jail.

Likewise, if they have chatlogs, witness testimony or other evidence that states you have a hidden volume on your drive, there may not be much that VeraCrypt can do to protect you.

A hidden volume within an encrypted partition or hard drive

If you have a hidden volume within an encrypted partition or hard drive, the forensic examiner will see a bunch of suspicious random data and probably assume that it is encrypted.

But if you stick to your "I securely erased the old data with VeraCrypt" excuse, it may just be enough to get you off. However, there is still a chance that you may not convince the authorities. This is especially true if there is other evidence that points to the existence of encrypted data.

If that excuse doesn't work, and the authorities plan to throw you in jail, you do have your backup plan. You can give them the key to the outer volume, which hopefully has some decoy files (it should) to throw them off. This might be enough to get them off your tail.

However, if the authorities are already suspicious because of your first lie, they may not be satisfied with the answer. Forensic examiners aren't stupid, and they know what hidden volumes are.

This leaves the chance that even your protective outer volume isn't enough to keep you out of jail. This becomes dramatically greater if there is other evidence that points to your hidden volume.

A hidden operating system

If you have a hidden OS on your device, it's likely that you have the VeraCrypt boot loader on it, or on a recovery disk nearby, otherwise you would never be able to use it. For this reason, you should assume that the authorities will see the boot loader and know that you use the program.

If you are using VeraCrypt as recommended and have set up a decoy operating system properly, you can hand over the key for the decoy to the authorities. The forensics expert can then check it out, see that it has been in use, and that it has seemingly legitimate files.

They will know that you use VeraCrypt, and also be able to see signs of the outer volume on a second partition that houses the hidden operating system. The examiner will want the password to the outer volume as well, but if you have some semi-sensitive looking files in there, it may be enough to allay their suspicions.

While there won’t be any damning evidence of the hidden operating system, there is no guarantee that the setup will be enough to keep the authorities off your back. There are some plausible reasons to have two VeraCrypt encrypted partitions on a single drive, but they are pretty unusual.

This could convince the investigators that there is a hidden operating system. If this hunch is compounded by other evidence, your likelihood of getting away with it may become more complicated.

Will plausibly deniable encryption save you from key disclosure laws?

There are so many variables, that it’s hard to say for sure. The jurisdiction, the severity of the crime, how the judge or prosecutor are feeling, and more. If you haven’t used VeraCrypt properly, or there is other evidence, then it probably won’t keep you safe from being thrown in jail for contempt.

If everything is above board and all the authorities have are hunches – even if they feel that they are almost certain – then deniable encryption may be your savior. It seems like it would set a dangerous precedent to send someone to jail for something that the authorities can’t even prove exists.

If suspects were held in contempt on the assumption they were using deniable encryption, it would probably lead to wrongful convictions for those rare people who do use tools like VeraCrypt to wipe their hard drives, or have setups that raise suspicions.

While there are no guarantees that plausibly deniable encryption can keep you safe from key disclosure laws, it probably can’t hurt.

But wait, couldn’t you just say you forgot the password, instead of going to all of the hassle of setting up deniable encryption?

Is it easier to say you forgot?

In our research, we mainly found cases of people being jailed for refusal to hand over their passwords or on other legal technicalities. Even in the one case where it seems that someone was jailed over forgetting their password, we don’t have enough detail to be certain.

There have been many cases where people have refused to hand over their passwords and escaped charges through the Fifth Amendment or other protections. A recent example occurred in 2019, when Pennsylvania’s highest court ruled that the Fifth Amendment prohibits forcing suspects to hand over their passwords to police.

This is despite a very clear admission of the evidence contained on the hard drive:

“*We both know what’s on there. It’s only going to hurt me. No f****** way I’m going to give it to you.”*

Some of those who have refused to hand over their passwords and were punished include:

- In 2009, a man was sentenced to nine months imprisonment in the UK under the Regulation of Investigatory Powers Act. He had refused to hand over the keys for several hard drives in relation to charges that were subsequently dropped.
- In 2010, a UK man was sentenced to 16 weeks for refusing to give police the password to an encrypted file on his computer. This is considered an offense under the Regulation of Investigatory Powers Act.
- In 2017 a UK judge gave a man a conditional 12-month discharge and an $830 fine for refusing to hand over his password.
- In 2018, a UK man was sentenced to 14 months in jail for failing to comply with an order under the Regulation of Investigatory Powers Act. He was ordered to disclose his Facebook password in relation to another crime that he was suspected of, but refused to provide the password.

- In 2019, a US man was asked by officers for the passwords to his phones after an arrest. He refused, but was handed warrants for the codes while out on bail. He refused again and was ordered to be locked up for contempt of court. This resulted in him spending 44 days in jail before the charges were thrown out.

These cases, plus others collated by the Electronic Frontier Foundation show that courts can go either way when it comes to refusing to hand over your password. If the judge rules against you, you can face heavy penalties.

So what if you try the old “I forgot” trick?

Philadelphia man jailed for contempt

A Philadelphia man suspected of various crimes initially pleaded the Fifth when authorities sought to compel him to disclose the keys for two hard drives and a phone.

He was directed by the Magistrate Judge to comply with a Decryption Order and decrypt the drives and the phone. He entered the correct password for the phone, but when it came to the drives, he entered several passwords, and none of them worked.

He claimed that he could not remember the passwords for the hard drives, which resulted in the Magistrate Judge granting a motion that ordered him to show cause. Based on the evidence presented at the hearing, the Judge found that he remembered the passwords but was willfully disobeying the order.

At a hearing before the District Court, the man did not testify, call any witnesses, offer up any evidence or explain why he failed to comply with the order. He was then remanded in custody until he fully complied with the Decryption Order.

He was jailed for four years until February 2020, when he was freed based on an appeal to a three-judge panel of the 3rd Circuit, which ruled that confinement based on contempt should not exceed 18 months.

The reason we waded so deeply into this legal quagmire, is that it’s easy to take single lines from it, such as that he ”…stated he could not remember the passwords…” and presume government overreach – that the US is now jailing people for forgetting their keys.

When you read the documents and look at the case in its full context, it’s not so straightforward. Sure, he claimed to have forgotten the passwords, but that was only after initially pleading the Fifth. Yes, he was held in custody for contempt of court, but this is because he “… offered no on-the-record explanation for his present failure to comply.”

He did not officially defend himself in the District Court using the excuse that he forgot his passwords, so we cannot infer from it that the courts are jailing people for forgetting their keys.

Florida man jailed for six months

A man from Florida was arrested on other charges and had his phone seized. Some time later, he provided the authorities with a password, however, it did not unlock the device. The man swore under oath that he had given the authorities the password at the time.

However, because the password didn’t work, the man was jailed for six months for contempt of court.

Details of the case are scant, but the implication seems to be that the defendant was claiming under oath that the password he had given was correct, and that either the authorities mixed it up, or he was misremembering it.

Without a full review of the court documents, it’s hard to get to the bottom of it, but he may have been jailed for refusing to hand over the password, or for forgetting it.

Florida judge lets collaborators off

In the very same state, things ended differently for two people charged for their collaboration on a crime. When the court demanded their passcodes for their phones, both suspects said that they forgot. In two separate rulings, the judges accepted that it was plausible for both suspects to have forgotten their codes.

At the time of the ruling, the phones had been seized more than a year beforehand, with one judge commenting, “There’s nothing jumping off the page that she’s willfully lying to me.”

Is forgetting good enough?

It seems like forgetting can be an acceptable excuse, especially if you stick to your story and there is no evidence to point out otherwise. It’s not exactly simple to tell the difference between someone who legitimately forgot and someone who is withholding information.

While saying that you forgot may be good enough, it’s not something that you can be certain will keep you safe. Plausibly deniable encryption may be a better choice, because it gives you another layer of protection.

Even if they did manage to get you to admit you had a hidden volume, you could still try the “I forgot” excuse when you got to that stage. However, the authorities may be less inclined to believe you once they are aware of your previous deception.

Scenario 3: Your adversary is technically proficient and may go to extreme lengths

This applies to authoritarian regimes and countries where the rule of law is not followed, places where there is no due-process, where draconian sentences could be imposed and torture or violence may be used against you.

It’s also relevant if your adversary has no authority over you, but may resort to extralegal means like violence. Things like sophisticated criminal gangs, highly organized extortion schemes, or even a technically proficient abusive partner.

In these cases where your adversary has the technical know-how and understands computer forensics and tools like VeraCrypt, plausibly deniable encryption could end up causing you even greater harm.

When these adversaries see the signs of deniable encryption, they will become suspicious. If you are incredibly lucky, you may be able to reason with them through one of the plausible excuses we have discussed.

But what would make them believe you? If they don’t care about your rights, fairness or inflicting harm on others, then why not torture you or lock you up forever, just in case you have extra information you were hiding?

VeraCrypt can make these situations especially dangerous, because you can never prove that you have given up all of the information and that there are no more hidden volumes on a device. Remember how it encrypts the free space as well, so everything looks the same?

This could result in extra torture or a continued sentence, even when you have handed over every piece of information you have.

Alternatives to deniable encryption

These are some of the most complicated situations to protect yourself and your data from. One of the few options is to use secret sharing, and have the data only accessible when the keys from two or more people are entered into the system.

For this to work, you would have to widely publicize that the information is only available when multiple people enter their passwords, in a similar way to how banks have signs about their time-delay safes that cannot be opened without waiting, no matter what. If your adversary doesn't know that this system is in place, they may end up torturing you or imprisoning you because they don't believe you.

Another solution is to set up the data to self-destruct if a certain key is entered, using the Kali Linux feature. If you are captured by your adversary, you can simply give them the self-destruct key, and the data from the device will be unrecoverable. However, this method only works if your attacker didn't have the foresight to copy the data first, which any technically proficient adversary should do.

If you do this, things will probably end up very badly for you. By deleting the data, you will have made your adversary extremely angry, and severe torture, long incarcerations or death aren't out of the question.

This technique is only advisable if you are in a situation where keeping the data from falling into the adversary's hands is more important than your life.

Scenario 4: Your adversary is not technically proficient but is willing to go to extreme lengths

If your adversary doesn't have much technical knowledge, there is a very good chance that you will be able to fool them with your hidden volumes or operating systems. If they can't see the data they are looking for, and everything seems to check out according to their basic level of knowledge, your plausible deniability may work.

In these situations, plausible deniability is probably an effective defense mechanism.

Scenario 5: You are traveling into a country where the authorities can force you to hand over your key at the border

Many countries can legally force people to hand over their devices and passwords when they cross borders. If you refuse to hand them over, you may be denied entry, or face a lengthy detainment.

Plausibly deniable encryption may enable you to safely hand over your data and a password to a decoy volume or operating system while still protecting your sensitive files. However, it all depends on the circumstances.

If the border guard isn't technically proficient, then this may fool them. However, it's reasonable to assume that they have skilled people involved in the process, if they are going to all the effort of looking through people's devices.

If so, the random data or VeraCrypt boot loader may be enough to set off alarm bells, causing you to be detained and possibly miss your flight, or worse.

A better option is to wipe your device with a secure, but less suspicious tool than VeraCrypt, and then restore it from the cloud when you arrive at your destination.

This allows you to hand over your device and password with less fear of being detained, because the border agent can see that there is nothing there.

Does plausibly deniable encryption really work?

The answer is a disappointing “kind of”.

If you live in a country where you can be compelled to hand over your password, then it may end up protecting you. Similarly, it could help to keep your data safe against a criminal with limited tech knowledge. But when facing authoritarian regimes or skilled people who are willing to torture you, it may end up harming you further.

If you live in a country where you can’t be legally forced to give up your keys, and your threat model doesn’t include violent adversaries, then plausibly deniable encryption is unnecessary. When crossing borders, there are better alternatives.

Honestly, for the vast majority of people who aren’t activists or doing anything illegal, simply encrypting your data with a strong password should be enough to protect it and yourself in 99.99999 percent of situations.

Because of the low chances of ever being in a situation where plausibly deniable encryption could help them, these people would probably be better off dedicating the time and effort to reducing risks of harm in other aspects of their lives.