Truly deniable encryption

onetuseronetuser 28122 silver badges66 bronze badges ⋮ 8-10 minutes

---

Disclaimer: I'm the author of `cryptsetup-deluks` and `grub-crypto-deluks`.

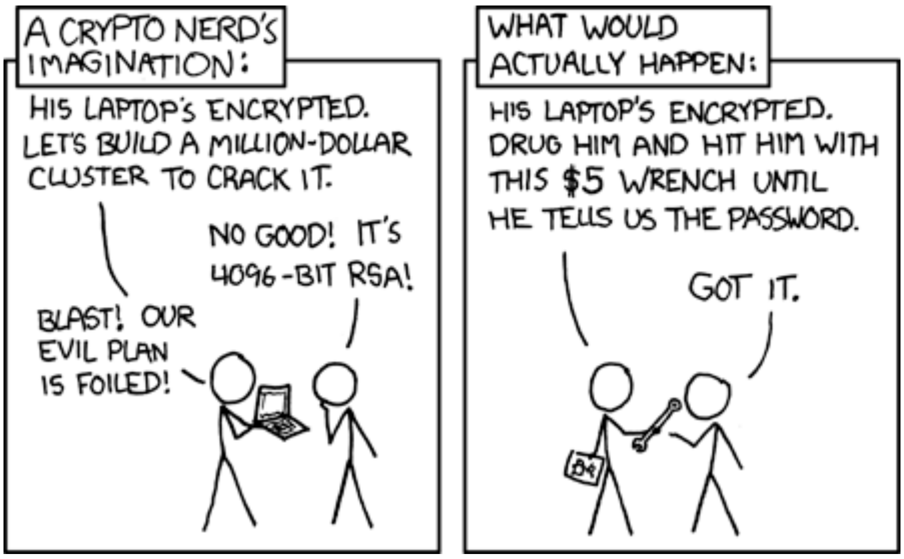
Deniable encryption is only a **part of the solution**. There's no perfect solution to protect yourself and your data if you get caught by an adversary. You can hardly avoid the suspicion of encryption, even if the adversary can't prove it. I'll list the issues and solutions.

Short partial answer

- Truecrypt header: `Veracrypt` (Windows/Linux/MacOS) but beware the bootloader
- Plain-mode encryption: `grub-crypto` (Linux) + plain-mode `cryptsetup`
- DeLUKS header: `grub-crypto-deluks` (Linux) + `cryptsetup-deluks`

Long complete answer

Defeating torture


A CRYPTO NERD'S IMAGINATION:
HIS LAPTOP'S ENCRYPTED. LET'S BUILD A MILLION-DOLLAR CLUSTER TO CRACK IT.
NO GOOD! IT'S 4096-BIT RSA!
BLAST! OUR EVIL PLAN IS FOILED!
WHAT WOULD ACTUALLY HAPPEN:
HIS LAPTOP'S ENCRYPTED. DRUG HIM AND HIT HIM WITH THIS $5 WRENCH UNTIL HE TELLS US THE PASSWORD.
GOT IT.

Torture in this case is also known as rubber-hose cryptanalysis. The good news is that torture is often inefficient. Your resistance to torture is key, or at least your ability to provide unverifiable fake facts (like the key to decrypt a decoy O.S.). Ideally, to stop torture, you should also be able to prove that you can't decrypt the data.

Deniable encryption doesn't solve the rubber-hose problem.

Solutions fall in 2 categories. See Technology that can survive a “Rubber-Hose attack” for details.

1. **You don't have the key:** the key could be split and its parts owned by several friends or hidden servers, acting as authenticators to check that you are not in a rubber-hose situation.
2. **The key was destroyed:** self-destructed (alarm system) or manually destroyed (chompable microSD card, etc). Harder to put in place and time-critical, but more resistant to yourself after torture.

Hiding the encryption header

Don't use an (unencrypted) LUKS header, it's not deniable.

Solutions:

- **Plain encryption:** *(plain dm-crypt, losetup, etc)* DON'T use it unless you have a very strong password. It has no protection against brute-force attack, rainbow-table attack and the password can't be changed unless you re-encrypt the whole payload data. Something you can do to harden plain encryption is to use the password to decrypt a given sector, and use this decrypted sector as the key to decrypt the payload data; but the drawback is that you have to remember a randomly-chosen sector number. The recommended Linux tool for plain encryption is `cryptsetup --type plain` as it is maintained.
- **Truecrypt/Veracrypt:** Truecrypt header is deniable, especially when used as full disk encryption. Truecrypt uses a salt and a fixed number of password hash iterations (PBKDF2) to decrypt the header containing the key so it's not prone to rainbow-table attack and considerably slows down brute-force attacks. However, the fixed number of hash iterations makes it less evolutive against brute-forcing than LUKS (as computing power increases, the number of password hash iterations should increase). Compared to LUKS, Truecrypt has no support for multiple keys and no GRUB (Linux bootloader) support.
- **DeLUKS:** A Deniable version of LUKS, for Linux. Mostly like Truecrypt, but with a support for GRUB bootloader, cryptsetup, multiple keys and an evolutive number of hash iterations.

Hiding the encrypted payload

Of course, you wouldn't put your data in a file called `Hey I'm encrypted.bin`. But even with full disk encryption, the simple existence of random-looking data could be used against you. See: In The UK, You Will Go To Jail Not Just For Encryption, But For Astronomical Noise, Too.

It's worth to note that, whatever the solution, there must be at least 1 decoy (fake) O.S. on your computer to not look totally suspicious.

Solutions have various levels of credibility and none is perfect. So, how do you explain this random-looking data?

- **Manually generated random data:**
  - **Disk wipe:** That's the n°1 explanation for a fully-encrypted drive. But is it sufficient?

    - **Noise recording**: It could be a file containing astronomical noise. You'd better have a capture device.
    - Etc.
- **By-default software behavior:**
    - **Truecrypt/Veracrypt hidden volume:** Truecrypt always wipes a newly created encrypted volume payload with random data. So it leads to a suspicion about the existence of a hidden volume in the Truecrypt container, that can't be proven under good circumstances.
- **Steganography:** This is the practice of concealing a file, message, image, or video within another file, message, image, or video. Impractical for system encryption, so it's better used for transmitting messages rather than storing them.
- **Hardware solutions:**
    - **Hardware-encrypted drives:** The presence of random data could be due to the manufacturer initial wipe. But these drives may not be optimized to be used in a deniable encryption way.
    - **Low-level data storage:** We could imagine to write data to the drive with a low magnetic level, near the noise level, to make the data appear as 0 under normal circumstances. I don't know any proof of concept of this.
- **"Cloud"-stored data**: Not really practical if it doesn't fit in your RAM size and needs to be written to your local disk. Moreover, it can be subject to a block cipher mode attack if the adversary has access to the encrypted data.

The not-so-deniable bootloader

System encryption is key to protect your data. Don't forget that your RAM is sometimes written to the on-disk swap and that your system keeps logs and tracks of opened files, etc. So it's advisable to use system encryption, and a bootloader that supports it.

At this time, all major O.S. bootloaders don't support natively plain encryption or deniable encryption headers. So the presence of a bootloader supporting deniable encryption can be used against you.

The known bootloaders with deniable encryption support are:

- **Veracrypt bootloader**: Plausible, you think?

  HEY, I'm an encrypted system, and trust me, there's no hidden volume!

- **`grub-crypto`**: *(a bootloader that supports plain-mode encryption)* At least the GUI looks like GRUB but the binary code betrays it.
- **`grub-crypto-deluks`**: *(grub-crypto fork with DeLUKS support)* Same as grub-crypto.

Solutions:

- **Decoy deniably-encrypted O.S.:** You explain the presence of the bootloader because there is another, decoy, encrypted O.S. And you only give the key to open this decoy O.S.
- **Loading the initramfs from a Live O.S.:** I don't know if it's even feasible. The idea is to use `cryptsetup` in plain-mode from a Live CD to decrypt the real O.S. Then you load the initramfs from the Live O.S. (and not from GRUB) to boot the real O.S.

- **Bootloader on a stick:** The bootloader with deniable encryption support is put on a USB stick. I don't believe in this but some people believe it is safer. It supposes your USB stick won't be seized along with your computer.

Currently inexistant solutions:

- **Deniable encryption support in mainstream bootloaders**: Unless you're a GRUB developer, you can just vote on the 2+ years old grub pull requests to merge `grub-crypto`.
- **Toolbox bootloader:** It would be helpful to have a toolbox bootloader (like GRUBS Reanimated USB Boot Stick) with, among other functionalities, the deniable encryption support. The presence of this special bootloader version would be explained by the other functionalities.

About network activity

Extract from Veracrypt: Security Requirements and Precautions Pertaining to Hidden Volumes:

> The computer may be connected to a network (including the internet) only when the decoy operating system is running. When the hidden operating system is running, the computer should not be connected to any network, including the internet (one of the most reliable ways to ensure it is to unplug the network cable, if there is one). Note that if data is downloaded from or uploaded to a remote server, the date and time of the connection, and other data, are typically logged on the server. Various kinds of data are also logged on the operating system (e.g. Windows auto-update data, application logs, error logs, etc.) Therefore, if an adversary had access to the data stored on the server or intercepted your request to the server (and if you revealed the password for the decoy operating system to him), he might find out that the connection was not made from within the decoy operating system, which might indicate the existence of a hidden operating system on your computer.