Cameleonica
safe cryptographic steganographic advanced filesystem

Arkadiusz Bulski

Abstract

This paper explains what the mission statement is for the project, the end-goals to be ultimately achieved. Purposefully omitted are any technical details and means to ends. Overall project is mostly described through basic definitions and examples. Use cases showing obvious benefits are also presented.

Introduction

> *Cameleonica is a safe cryptographic steganographic advanced filesystem.*

This succinct definition best describes the overall project (and is the official description for the project on GitHub). That is however exactly because it uses loaded terms. Each of the terms carries a very elaborate meaning which will be discussed below. Above statement can also be summarized as:

> *It aims to guarantee confidentiality, authenticity, plausible deniability, transactions, snapshots and versioning, instantaneous copying, permanent deletion, high performance and low delays, compression, and hashing.*

These properties are more elaborately explained through definitions and examples:

Filesystem is a data structure that maps a hierarchy of regular files and directories into a flat file that acts as storage. Access to the files is governed by rules that make up so called semantics. Most semantics are already well defined by POSIX standard. The fact that open files can be removed and replaced while still being open is a notable example. Linux man-pages define even more fancy semantics. For example, `fallocate` allows to remove a range of bytes without leaving a hole, stitching together both sides. For another example, `creat` can create a nameless file that can later be renamed and thus saved. Replacing files is an atomic action, often used as means of ensuring one of the two versions remain on disk. All these semantics are interesting but are not the selling point for Cameleonica. They only build a foundation for general purpose filesystem.

Cryptographic filesystem is a filesystem that provides two attributes: confidentiality and authenticity. Confidentiality is a property that easily translates to filesystems. If a

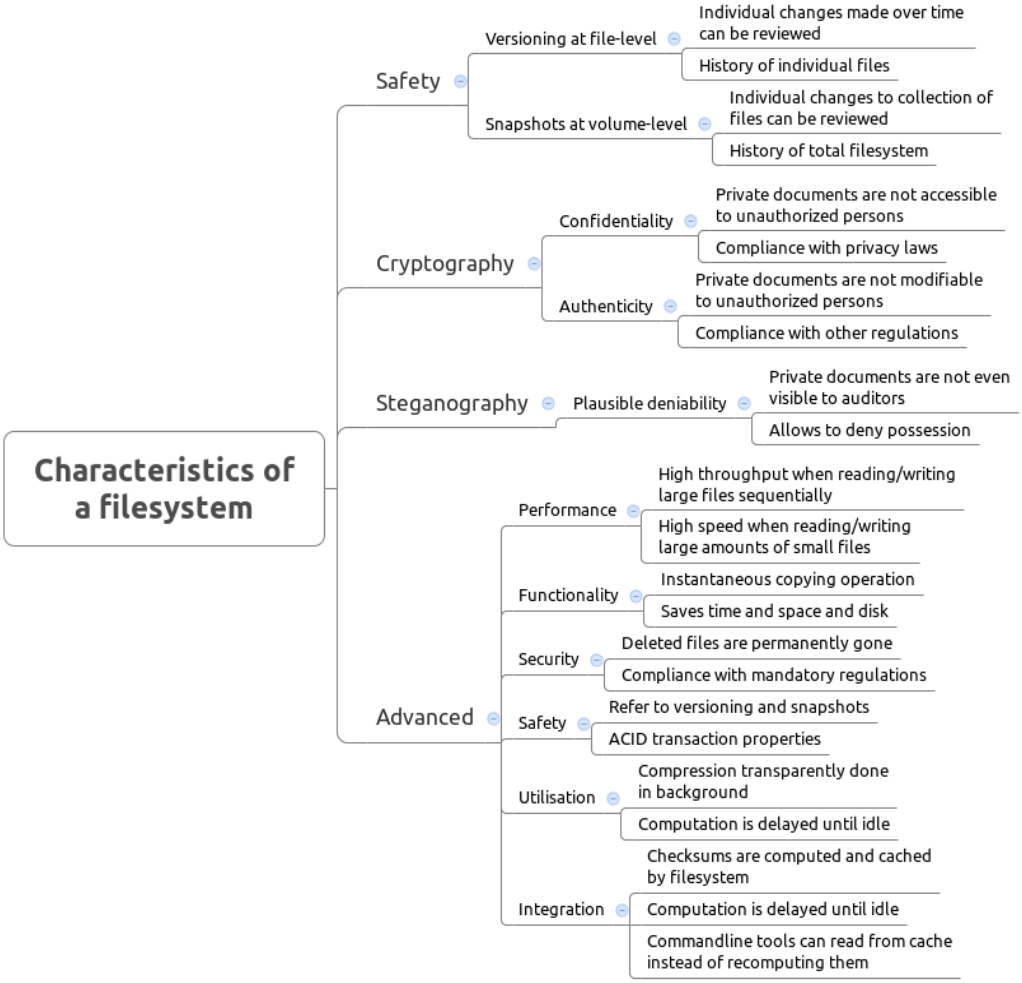
filesystem is encrypted, it should not reveal any information about it's file structure, file names, sizes or content until a valid password is provided. Authenticity is a property that guarantees that the files read now are the files written earlier. That is, a valid password must be provided before any changes can be made to the file structure. No file can be moved, truncated, or it's content changed without a valid password. Random, unintelligible changes to files are not allowed to remain undetected. If bytes being read were modified without a valid password, an error must be returned upon read. It is not allowed for read to return garbage bytes in this situation.

Steganographic filesystem is a filesystem that provides more than one file structure using only one backend storage. Every file structure is encrypted with an independent valid password. Steganographic confidentiality is a property that demands that if one valid password is revealed, remaining valid passwords do not get revealed. That is, existence of other valid passwords remain concealed, on top of their corresponding file structures remaining encrypted. Steganographic confidentiality is a stronger notion than cryptographic confidentiality. Multiple independent valid passwords can exist at same time. Cryptography usually deals with only one password at a time, as far as human interaction goes. More importantly, the mere existence of that one password is hardly a secret. In steganography, one password must necessarily not lead to discovery of other valid passwords. Cryptography is concerned with keeping the content concealed, not with concealing the content's very existence.

Safe filesystem is a filesystem that refuses to become unusable or loose access to already established content in event of abrupt interruption. Modern filesystems are able to reliably recover after an unexpected power loss encountered at any point in time. This safety guarantee comes without any extra settings turned on. It is commonly expected from any popular filesystem to handle interruptions gracefully. However, commonly established semantics are far from being acceptable. Once a file was opened for writing, an interrupted write can be partially successful, leaving content neither in the state before opening, neither in the state expected after a complete write. This outcome is completely unacceptable. Content from before changes started should be available for recovery. Another scenario is when a user copies a file while overwriting the destination. Old destination file gets truncated to zero before new content is copied into it, which may take several minutes. After interruption, user can expect old content to be gone while new content is only partially present, ending at random location. This outcome is also unacceptable. Old content from before truncation should be available for recovery. Another scenario is when a program modifies a series of files in a related way. Rotating pictures in a photo album can serve as an example. User could not only expect a rotation to be atomic, but could also expect the whole album to be processed seemingly at once. After interruption, user could demand the whole album to be reverted to original state. Mentioned problems are not new and can be solved using existing mechanisms. Snapshots are becoming quite popular lately, however they are too cumbersome to be of practical use. They have to be taken manually, often enough, and usually cover a whole volume at once. This is clearly not the right way to go. Snapshot infrastructure should retain the state of file structure after every significant change, automatically and transparently. Usable recovery scenario should be easily discoverable for any of mentioned scenarios.

Advanced filesystem is a filesystem with outstanding usability. Usability can be meant as performance (close to maximum speeds of underlying hardware), as functionality (instant copying), as security (deletions equivalent to permanent wiping), as safety

(versioning, snapshots), as utilisation (transparent compression), as integration (speeding up utilities). Performance of modern filesystems is getting close to physical limitations of underlying hard-drives. Processing of big files would be expected to be happening at throughputs close to possible maximum. Processing of small files would be expected to be happening without a delay due to write-back caches. Functionality of modern filesystems is mostly ancient. Paradigm of opening a file for writing, then changing it one fragment at a time, has not changed since practically ever. There is still room for change tho. Hard-links were invented as means of instantaneous forking of files. This however only works in a read-write fashion. It is possible to achieve similar characteristics for copying operation, resulting in a read-only fork. This is functionally equivalent to single file versioning. Secure deletion is another area where modern filesystems are lacking. Wiping files is not practical. User has to manually take action, and if he fails to do it right, unwanted evidence remains basically forever. File removal could be made quick and permanent using basic cryptography. Versioning is another functionality that is commonly missing. Risk could be minimized by underlying filesystem by automatically savepointing after every change, regularly savepointing for later review, and manually savepointing at user discretion. Disk utilisation can be increased through compression of selected files. Compression can happen transparently, without distracting the user, and with delay, without the user waiting for addition computation to be done. Hashing can also be speeded up when using common utilities. Filesystem can compute file checksums and provide them to system utilities when they are about to compute it themselves. Computation can also be delayed, computed in advance when system is idle and cached for future use. System utilities can be modified to take advantage of cached checksums.



Use cases

People with different needs can benefit from using a versatile filesystem. There are several reasons why one might want to use certain functionality. Provided below are possible scenarios of people benefiting from different subsets of functionality:

Graphic designer. He works at an advertising company. Everyday he edits images and photographs provided by his agency. At regular meetings he exchanges materials with his coworkers, where everybody copy source materials from anybody else and also copy their produced work to their superiors drives. Quick processing of medium sized files is beneficial, as it regularly saves time. After morning meeting, he started working on his last assignment. He would like to browse history of his changes to see what remains to be done. Regular snapshots and individual file versioning makes reviewing his work easy. He can now resume his work. While editing an image, his editor crashed. He would like to revert to a state few changes back but the image was saved after they were made and there is no undo possible after editor was closed. Filesystem keeps a continuous history of recent changes and fine grained undo is possible. He can revert to a state from after last save, and the save before, and the one before that. After a day worth of work, he copies his current work into project archive overwriting yesterdays version with todays. Right after he started copying files power went off. After power was restored, old version was gone and new version was only partially copied. Replacing operation triggered a savepoint however, and old version was quickly recovered. New version was then copied again and finished successfully.

Human rights activist. She works for a local newspaper. Country in which they operate is ruled by an oppressive regime. Her daily job activities revolve around gathering incoming reports and writing articles to be printed. She drives a lot around the country and encounters random checkpoints. She must protect her sources and cannot allow her notes to be seized by an opportunist militia solder. If randomly searched, her laptop is encrypted and does not provide access to unauthorized persons. She refuses to decrypt it until a warrant is presented and her agency lawyer arrives. She is briefly questioned about her business and let go. After getting back home she gets detained by a security force and questioned about any involvement with anti-government organizations. She admits having no involvement. She is presented with a warrant to search her computers. She agrees to comply and provides a valid password that decrypts some documents on her laptop. To the auditors it is clear that the provided password is indeed valid and she complied. Her laptop is throughly checked and only expected agency documents are discovered. Her interrogators do not stop accusing her of being a suspected member and keep searching her laptop for incriminating evidence. Indeed, she was regularly in contact with rebel forces and her laptop contains illegally obtained documents. If these documents were found, or even a hint of their existence was found, she would likely be taken to jail. Ultimately, no evidence is found on her computer showing that she hides any documents and she is let go.

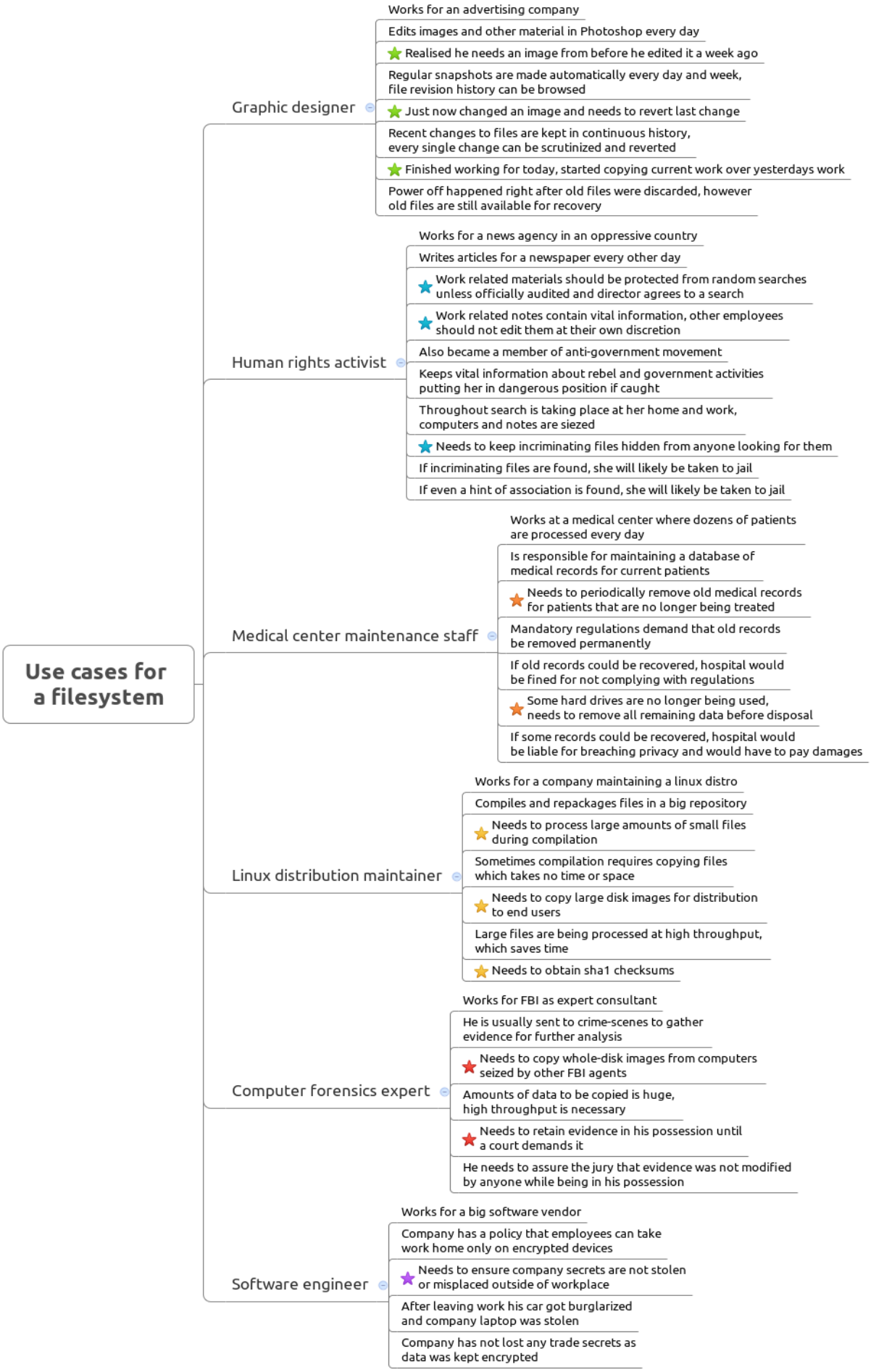
Medical center maintenance staff. He works at a major hospital that processes dozens of patients every day. His job is to maintain a database of medical records of current patients. Regularly he has to delete old records of former patients. Government regulations demand that these records are permanently purged. Medical center has an

obligation to guard privacy of it's patients and keep their records confidential. If these records would resurface later the center would get fined for breaking regulations. He can rest assured that deleted files are permanently gone, as the filesystem guarantees it. Aside of regular tasks, there is a need to retire some hard-drives that were used for years. He needs to remove any remaining files from them before he can send them for disposal. Again, data need to be purged permanently or the center would be liable and would have to pay damages. He can rest assured that formatting drives destroyed everything permanently as the filesystem guarantees it.

Linux distro maintainer. He works for a company maintaining a linux distro repository. Everyday he compiles and archives whole collections of source code and binaries. When he compiles, scripts often copy files which actually takes no space and no time. This takes away some time from compilation time. After a day worth of work, he sends a big disk image with upgrades. Big files are being copied at high throughput, which again saves time. Afterwards he needs to obtain sha1 checksums. Filesystem computed checksums already during copy operation. When he uses a standard command-line utility to obtain a checksum it gets results immediately from cache.

Computer forensics expert. He works for FBI as a consultant. He is often being sent to crime-scenes to secure evidence. When a computer gets seized his job is to make disk images of confiscated hard-drives. Disk images take a huge amount of space. High throughput is necessary to make copies in acceptable amount of time. After copies are done, he needs to keep the copies in his possession until later. Court later demands evidence to be presented and jury needs to be assured that these copies were not modified after being obtained. The expert was the only user with a password and filesystem can guarantee that noone else could write to the filesystem.

Software engineer. He works for a major software vendor. His company has a policy that employees can bring work home only on encrypted devices. After work he took a copy of current project on a company laptop and drove back home. When shopping, his car got burglarized and the company laptop was stolen. Filesystem was encrypted and he can be sure that no company secrets fell into wrong hands. Company executives are relieved that there was no major loss and a new laptop was issued to the employee.


Use cases for a filesystem
Graphic designer
Works for an advertising company
Edits images and other material in Photoshop every day
Realised he needs an image from before he edited it a week ago
Regular snapshots are made automatically every day and week, file revision history can be browsed
Just now changed an image and needs to revert last change
Recent changes to files are kept in continuous history, every single change can be scrutinized and reverted
Finished working for today, started copying current work over yesterdays work
Power off happened right after old files were discarded, however old files are still available for recovery
Human rights activist
Works for a news agency in an oppressive country
Writes articles for a newspaper every other day
Work related materials should be protected from random searches unless officially audited and director agrees to a search
Work related notes contain vital information, other employees should not edit them at their own discretion
Also became a member of anti-government movement
Keeps vital information about rebel and government activities putting her in dangerous position if caught
Throughout search is taking place at her home and work, computers and notes are siezed
Needs to keep incriminating files hidden from anyone looking for them
If incriminating files are found, she will likely be taken to jail
If even a hint of association is found, she will likely be taken to jail
Medical center maintenance staff
Works at a medical center where dozens of patients are processed every day
Is responsible for maintaining a database of medical records for current patients
Needs to periodically remove old medical records for patients that are no longer being treated
Mandatory regulations demand that old records be removed permanently
If old records could be recovered, hospital would be fined for not complying with regulations
Some hard drives are no longer being used, needs to remove all remaining data before disposal
If some records could be recovered, hospital would be liable for breaching privacy and would have to pay damages
Linux distribution maintainer
Works for a company maintaining a linux distro
Compiles and repackages files in a big repository
Needs to process large amounts of small files during compilation
Sometimes compilation requires copying files which takes no time or space
Needs to copy large disk images for distribution to end users
Large files are being processed at high throughput, which saves time
Needs to obtain sha1 checksums
Computer forensics expert
Works for FBI as expert consultant
He is usually sent to crime-scenes to gather evidence for further analysis
Needs to copy whole-disk images from computers seized by other FBI agents
Amounts of data to be copied is huge, high throughput is necessary
Needs to retain evidence in his possession until a court demands it
He needs to assure the jury that evidence was not modified by anyone while being in his possession
Software engineer
Works for a big software vendor
Company has a policy that employees can take work home only on encrypted devices
Needs to ensure company secrets are not stolen or misplaced outside of workplace
After leaving work his car got burglarized and company laptop was stolen
Company has not lost any trade secrets as data was kept encrypted