Comparison with Other Projects - gocryptfs

9-11 minutes

There are several open-source file encryption solutions for Linux available. In contrast to disk-encryption software that operate on whole disks (TrueCrypt, dm-crypt etc), file encryption operates on individual files that can be backed up or synchronised easily.

This page compares:

- gocryptfs (this project), aspiring successor of EncFS
- EncFS, mature with known security issues
- eCryptFS, integrated into the Linux kernel
- Cryptomator, strong cross-platform support through Java, WebDAV and FUSE.
- securefs, a cross-platform project implemented in C++. Older versions stored directories in user-space B-trees (filesystem format 1,2,3). The new default since v0.7.0 (filesystem format 4) uses normal directory entries.
- CryFS, result of a master thesis at the KIT University that uses chunked storage to obfuscate file sizes.

If you spot an error or want to see a project added, please file a ticket! See also: comparison table in the Arch Linux wiki

Overview

	gocryptfs v1.7	**encfs v1.9.5**	**ecryptfs v4.19.0**	**cryptomator v1.4.6**	**securefs v0.8.3**	**CryFS v0.10.0**
First release	2015 (ref)	2003 (ref)	2006 (ref)	2014 (ref)	2015 (ref)	2015 (ref)
Language	Go	C++	C	Java	C++	C++
License	MIT (ref)	LGPLv3 / GPLv3 (ref)	GPLv2	GPLv3 (ref)	MIT (ref)	LGPLv3 (ref)
Development hotspot	Austria	USA	USA (RedHat)	Germany	China	Germany
Lifecycle	Active	Maintenance	Active (ref)	Active	Active	Active
File interface	FUSE	FUSE	In-kernel filesystem	FUSE/WebDAV	FUSE	FUSE
Platforms	Linux, MacOS, 3rd-party Windows port cppcryptfs, 3rd-party Android port DroidFS	Linux, MacOS, 3rd-party Windows port	Linux	Linux, MacOS, Windows	Linux, MacOS, Windows	Linux, MacOS, Windows (experimental)
User interface	CLI, 3rd-party GUI (SiriKali)	CLI, 3rd-party GUI	Integrated in login process	GUI, 3rd-party CLI (ref)	CLI, 3rd-party GUI	CLI, 3rd-party GUI (SiriKali)
Reverse Mode	yes (since v1.1, read-only)	yes (limited write support)	no	no	no	no

General Security

	gocryptfs	encfs default	encfs paranoia	ecryptfs	cryptomator	securefs	CryFS
Documentation available	Yes [1]	Yes [2]	Yes [2]	No [4]	Yes [3]	Yes [5]	Yes [6]
Password hashing	scrypt	PBKDF2	PBKDF2	(none, implemented in external tool)	scrypt	PBKDF2	scrypt

References: [1] [2] [3] [5] [6]
[4] actually, there is a lot of ecryptfs documentation, but none of it seems to describe the used crypto.

File Contents

	gocryptfs	encfs default	encfs paranoia	ecryptfs	cryptomator	securefs	CryFS
Tested version	v1.7	v1.9.5	v1.9.5	v4.19.0	v1.4.6	v0.8.3	0.10.0
Encryption	GCM [1]	CBC; last block CFB [2]	CBC; last block CFB [2]	CBC	CTR with random IV [3]	GCM	GCM
Integrity	GCM	none	HMAC	none	HMAC	GCM	GCM
File size obfuscation	no	no	no	yes (4 KB increments)	no [4]	no [5]	yes (chunked storage)

References: [1] [2] [3] [4] [5]

File Names

	gocryptfs	encfs default	encfs paranoia	ecryptfs	cryptomator	securefs	CryFS
Tested version	v1.4.1	v1.9.2	v1.9.2	v4.12.5	v1.5.15 AppImage FUSE	v0.7.3-30-g2596467	0.9.7-15-g3d52f6a8
Encryption	EME [4]	CBC	CBC	CBC	AES-SIV	AES-SIV	GCM (dir DB)
Prefix leak	no (EME)	no (HMAC used as IV)	no (HMAC used as IV)	yes [2]	no (AES-SIV)	no (AES-SIV)	no (GCM)
Identical names leak	no (per-directory IV)	no (path chaining)	no (path chaining)	yes [1]	no [3] {3}	yes [6]	no (GCM)
Maximum name length [5]	255 (since v0.9) {2}	175	175	143	1024	143	1024
Maximum path length [5]	4095				4095		

	gocryptfs	encfs default	encfs paranoia	ecryptfs	cryptomator	securefs	CryFS
Directory flattening {1}	no	no	no	no	yes	yes	yes

References: [1] [2] [3] [4] [5] [6]

Notes:
{1} Is the directory tree flattened in the encrypted storage? This obfuscates the directory structure but can cause problems when synchronising via Dropbox and similar.
{2} 255 since gocryptfs v0.9, 175 in v0.8 and earlier
{3} cryptomator dropped the use of a random padding in v1.2.0 due to performance concerns.

Performance on Linux

All tests are run on tmpfs rule out any influence of the hard disk. The exact command lines for running the tests are defined in canonical-benchmarks.bash. The box that was used to running the tests has been upgraded with a new CPU {2}, and unfortunately not all tests have been re-run. Which CPU was used is noted in the table header.

	gocryptfs {2}	encfs default {2}	encfs paranoia {1}	ecryptfs {2}	cryptomator {2}	securefs {1}	CryFS {1}
Tested version	v2.3.2-3-g1a866b7	v1.9.5	v1.9.2	v6.2.13	v1.5.15 AppImage FUSE	v0.7.3-30-g2596467	v0.9.7-12-gd9634246
Streaming write	482 MiB/s	122 MiB/s	51 MiB/s	323 MiB/s	57 MiB/s	132 MiB/s	69 MiB/s
Streaming read	944 MiB/s	451 MiB/s	105 MiB/s	961 MiB/s	113 MiB/s	155 MiB/s	99 MiB/s
Extract linux-3.0.tar.gz	10.9 s	13 s	23 s	3.9 s	28 s	14 s	41 s
md5sum linux-3.0	5.1 s	5.7 s	10 s	1.2 s	15 s	7.7 s	42 s
ls -lR linux-3.0	2.0 s	2.5 s	2.9 s	0.5 s	4.3 s	1.2 s	17 s
Delete linux-3.0	2.4 s	3.4 s	4.4 s	0.7 s	10 s	2.2 s	21 s

Notes: {1} Tested on an Intel Pentium G630 with 2 x 2.7GHz that does NOT have AES instructions
{2} Tested in Intel Core i5-3470 CPU with 4 x 3.20GHz and AES-NI

Performance on Windows

All tests were run on a Toshiba-RD400 M.2/NVMe SSD rated 2.6 GB/s read and 1.6 GB/s write random-access speed. The operating system used was Windows 7 Professional SP1 running on an Intel Core i7-6700 CPU with 4 x 3.40GHz hyperthreaded and AES-NI. Tests were run using MSYS-CoreUtils 5.97-3-msys-1.0.13 installed using the MinGW installer. The exact command lines for running the tests are defined in canonical-benchmarks.bash with minor adjustments required to make the test run in this environment.

	(NTFS)	cppcryptfs	EncFSMP default	EncFS4Win default	cryptomator	securefs	CryFS
Tested version	v6.1.7601.24382	v1.4.0.25	v0.99.1	v1.10.1-rc14	v1.4.6	v0.8.3	v0.10.0.1201 {1}
Based on	-	gocryptfs 1.4 compatible	EncFS 1.9.5	EncFS 1.9.1	-	-	-
Driver	(Built-in)	Dokany 1.2.2.1000	PFM 1.0.0.192 {2}	Dokany 1.2.2.1000	Dokany 1.2.2.1000	WinFSP 2019.1	Dokany 1.2.2.1000
User Interface	-	GUI	GUI (fails to properly display state)	Tray (very basic)	GUI	No {3}	No {3}
Streaming write	2100 MiB/s {4}	621 MiB/s	58 MiB/s	68 MiB/s	67 MiB/s	289 MiB/s	51 MiB/s
Streaming read	3400 MiB/s {4}	797 MiB/s	251 MiB/s	107 MiB/s	115 MiB/s	542 MiB/s	130 MiB/s
Extract linux-3.0.tar.gz	26 s	456 s	793 s	2121 s	2497 s	332 s	1124 s
md5sum linux-3.0	51 s	364 s	235 s	1877 s	1808 s	235 s	1254 s
ls -lR linux-3.0	18 s	328 s	166 s	1269 s	1722 s	183 s	1057 s
Delete linux-3.0	18 s	432 s	(427 s) {5}	1666 s	2765 s	260 s	1007 s

To the extent this was observed at all during the tests, every one of these filesystem providers was fully CPU-bound during the small-file tests with observed disk access speeds never going beyond 15 MiB/s.

Notes:
{1} CryFS considered Windows support “highly experimental” in this version
{2} Closed source component by Pismo Technic Inc
{3} The SiriKali third-part GUI supports CryFS, EncFS4Win and securefs
{4} Yes, these numbers are actually above what the drive is theoretically capable of, so all of these results are likely somewhat skewed
{5} 320 files were not deleted due to *Invalid argument* errors; it is not clear what caused this error, but the logged “Invalid data size, not multiple of block size” messages may indicate corruption

Disk Space Efficiency

	ext4	gocryptfs	encfs default	encfs paranoia	ecryptfs	cryptomator	securefs	CryFS
Tested version	v4.12.5	v1.4.1	v1.9.2	v1.9.2	v4.12.5	v1.5.15 AppImage FUSE	0.7.3-30-g2596467	0.9.7-15-g3d52f6a8
Empty file {1}	0	0	0	0	8,192	88	16	32,768

	ext4	gocryptfs	encfs default	encfs paranoia	ecryptfs	cryptomator	securefs	CryFS
1 byte file {1}	1	51	9	17	12,288	137	45	32,768
1,000,000 bytes file {1}	1,000,000	1,007,858	1,000,008	1,007,888	1,011,712	1,001,576	1,006,876	1,048,576 {4}
linux-3.0 source tree {5}								
...disk usage {2}	494 MiB	512 MiB	495 MiB	498 MiB	784 MiB	520 MiB	498 MiB	1485 MiB
...sum of file sizes {3}	411 MiB	416 MiB	412 MiB	415 MiB	784 MiB	430 MiB	416 MiB	1485 MiB

Notes:
{1} `ls -l` on the encrypted file
{2} `du -sm` on the ciphertext dir, backing filesystem ext4.
{3} `du -sm --apparent-size`.
{4} Counting all 32 chunks (ref)
{5} Extracted linux-3.0.tar.gz

Filesystem Features

Note: To keep the work of maintaining this table under control, I have only tested selected projects with respect to filesystem features. Please file a pull request if you can test the other projects!

The backing filesystem is assumed to be ext4.

	ext4	gocryptfs	encfs default	encfs paranoia	ecryptfs	CryFS
hard links	yes	yes	yes	no	yes	no
extended attributes	yes	yes {1}	yes {2}	yes {2}	?	?
fallocate	yes	yes	no	no	no	no
fallocate KEEP_SIZE	yes	yes	no	no	no	no
fallocate PUNCH_HOLE	yes	no	no	no	no	no

Notes:
{1} Names and values encrypted
{2} Not encrypted

Previous Chapter Next Chapter