Как зашифровать файл или каталог в Linux?

Спросил 13 лет, 8 месяцев назад Изменено 3 года, 6 месяцев назад Просмотрено 62 тыс. раз

39

Какая самая популярная команда для шифрования файла или каталога в терминале Linux?

линукс шифрование

Делиться Улучшить этот вопрос Следовать

отредактировано 30 октября 2013 г. в 19:42

Scott - Слава Украины
22.1 тыс. 46 66 124

спросил 23 февр. 2011 г. в 18:30

пользователь2195

9 ответов

Сортировать по:

Наивысший балл (по умолчанию)

44

Я думаю, это будет GnuPG. Хотя синтаксис для файлов и каталогов отличается.

Шифрование

Для файлов (выходов `filename.gpg`):

```
gpg -c filename
```

Для каталогов:

```
gpg-zip -c -o file.gpg dirname
```

Расшифровка

Для файлов (выходов `filename.gpg`):

```
gpg filename.gpg
```

Для каталогов:

```
gpg-zip -d file.gpg
```

Обновление по устареванию

Кажется, `gpg-zip` команда устарела в последних версиях. Вместо этого используйте `gpgtar` команду или сожмите каталог (например, преобразуйте его в tarball), а затем зашифруйте его как файл.

Редактировать : Исправлено, поскольку @Mk12 указал на ошибку сжатия/распаковки для шифрования/дешифрования.

Делиться Улучшить этот ответ Следовать

отредактировано 9 мая 2020 г. в 18:33
МАКитгарха
125 7

ответил 23 февр. 2011 г. в 19:30
знаменитость
946 5 7

Разве это не должно быть «Шифрование» и «Дешифрование»? – мк12 27 июл 2012 в 18:41

1 Никто не сказал, как зашифровать каталог. – разведение 18 окт. 2014 г. в 8:50

1 @chovy Не то, что выше написано: Для каталогов: gpg-zip -c -o file.gpg dirname – знаменитость 20 окт. 2014 г. в 10:33

@celebdor пропустил это. Спасибо. правка: у меня это не работает. Я получаю какой-то странный зашифрованный вывод, когда расшифровываю файл. – разведение 24 окт. 2014 г. в 15:29

@chovy: Сожалею это слышать. Могу подтвердить, что шифрование и дешифрование каталогов, как показано выше, сработало у меня с использованием gpg-zip (GnuPG) 1.4.16 под Mint 17. – Майкл Шепер 12 июня 2016 г. в 9:13

13

- с OpenSSL

```
openssl des3 -salt -in unencrypted-data.tar -out encrypted-data.tar.des3
```

Расшифровать:

```
openssl des3 -d -salt -in encrypted-data.tar.des3 -out unencrypted-data.tar
```

- зашифровать с помощью AES

```
aescrypt -e -p password  file.jpg
```

Расшифровать:

```
aescrypt -d -p password file.jpg.aes
```

Делиться Улучшить этот ответ Следовать

ответил 23 февр. 2011 г. в 19:29
струя
2,733 19 17

1 +1 for showing how to do it with openssl, which is most likely available out-of-the-box. – DevSolar Aug 22, 2012 at 15:39

3 Indeed, but 3DES is considered insecure and should not be used, AES (aescrypt) is a much better option, see: stackoverflow.com/questions/1619212/… – jmng Jul 24, 2018 at 10:01

5

This is my method using openssl and tar

Open Encrypted Directory:

```
openssl enc -aes-256-cbc -d -in ~/vault.tar.gz.dat | tar xz; thunar ~/vault
```

Lock Encrypted Directory:

```
tar cz vault/ | openssl enc -aes-256-cbc -out ~/vault.tar.gz.dat; rm -r ~/vault
```

Share Improve this answer Follow

answered Jul 3, 2015 at 15:41
Tom
353 3 9

2 `rm -r` does not delete data; it merely unlinks it. You'll need to use something like `srm` to erase the data from the disk. – jbindel May 19, 2016 at 2:51

3

Try GnuPG.

To encrypt: `gpg -c filename`

To decrypt: `gpg filename.gpg`

Share Improve this answer Follow

answered Feb 23, 2011 at 19:28
slhck
232k 71 624 604

2

I personally use `aescrypt` mostly.

```
      aescrypt -e "File"
```

and decrypt:

```
      aescrypt -d "File"
```

Or there's mcrypt:

```
mcrypt "File"
```

and decrypt:

```
mcrypt -d "File"
```

And for a directory , I suggest tar'ing the dir, and encrypting that. Then after unencrypting, just untar the file:

```
tar -cf "Dir.tar" Dir/
```

and to untar

```
tar -xf "Dir.tar"
```

Share Improve this answer Follow

edited Aug 2, 2016 at 10:50
Community Bot
1

answered Aug 1, 2011 at 17:39
Matt
767 1 11 18

1

If highest level of security is not a big problem (the man page of zip says, that the encryption algorithm used by zipfile utilities are weaker than PGP), then I prefer zip and unzip. It zips my directories and encrypts at the same time. I prefer zip because you can have a kind of incremental zip and encrypt instead of zipping and encrypting the whole thing again. Especially it is useful when the directory sizes are very large.

ZIP and encrypt

```
zip file.zip file
zip -r directory.zip directory
zip --encrypt file.zip.enc file # prompt for password
zip --encrypt -r directory.zip.enc directory # prompt for password
```

Unzip and decrypt

```
unzip directory.zip.enc #Beware if any directory is present with the same name as the
zipped file, then it would be overwritten. Hence I normally send the contents to
another directory.

unzip directory.zip.enc -d directory-new # prompts for password
```

Share Improve this answer Follow

answered Aug 2, 2016 at 12:40
infoclogged
313 1 4 16

0

May not be popular but I've been working on a project to encrypt/decrypt anything with minimal user interaction through the use of a few Bash scripts. Here's a link to the Hak5 post that explains setup for testing.

Cutting through the source code logics though here's what happens for each type of data that can be handled by the above linked project

```
_gnupg_encrypt_opts="--always-trust --armor --batch --encrypt --recipient
user@host.domain"
 _bulk_output_dir="some_path"
_arbitrary_parsed_output="some_file.gpg"
## If file make encrypted time stamped file with similar name
_path_to_file="${_mapped_input}"
_path_to_output="${_bulk_output_dir}/$(date -u +%s)_${_path_to_file##*/}.gpg"
cat "${_path_to_file}" | gpg ${gpg _gnupg_encrypt_opts} > "${_path_to_output}"
## else if directory make compressed encrypted time stamped output file
_path_to_dir="${_mapped_input}"
_path_to_output="${_bulk_output_dir}/$(date -u +%s)_dir.tgz.gpg
tar -cz - "${_path_to_dir}" | gpg ${gpg _gnupg_encrypt_opts} >
"${_path_to_output}"
## else if something else append encrypted output to file
_path_to_output="${_arbitrary_parsed_output}"
cat <<<"${_mapped_input}" | gpg ${gpg _gnupg_encrypt_opts} >> "${_path_to_output}"
```

The `${_mapped_input}` variable is set by reading a `mkfifo` named pipe file and setting anything read to an array with `mapfile -t _lines < "${_file_to_map}"` which is later expanded and saved to a `${_mapped_input}` ... a bit convoluted but it allows for experimental features to act on individual lines. End results are you end up with a directory for holding encrypted files or compressed directories and a file with various packets of encrypted data.

Decryption for files or compressed directories is simple enough on a device with a private key related to the public key used for encryption. But decryption of multiple armor encrypted data packets was a bit tougher, so there a script named `Paranoid_Pipes_Scenario_One.sh` in the above project written to do it all with minimal user interaction. Below is a simplified version of the helper scripts source code for normal encrypted files and directories.

```
_gnupg_decrypt_opts="--quiet --no-tty --always-trust --passphrase-fd 9 --decrypt"
_decryption_output_dir="some_directory"
# if file
exec 9<"${_pass[@]}"
_path_to_file="${_mapped_input}"
_output_name="${_path_to_file##*/}"
_output_name="${_output_name%.gpg*}"
cat "${_path_to_file}" | gpg ${_gnupg_decrypt_opts} >
"${_decryption_output_dir}/${_output_name}"
# else if compressed file
_path_to_file="${_mapped_input}"
_output_name="${_path_to_file##*/}"
_output_name="${_output_name%.tgz.gpg*}"
mkdir -p "${_decryption_output_dir}/${_output_name}"
_old_pwd="${PWD}"
cd "${_decryption_output_dir}/${_output_name}"
cat "${_path_to_file}" | gpg ${_gnupg_decrypt_opts} | tar -xzf -
cd "${_old_pwd}"
# else if non-compressed directory
_path_to_file="${_mapped_input}"
```

```
_output_name="${_path_to_file##*/}"
_output_name="${_output_name%.tar.gpg*}"
mkdir -p "${_decryption_output_dir}/${_output_name}"
_old_pwd="${PWD}"
cd "${_decryption_output_dir}/${_output_name}"
cat "${_path_to_file}" | gpg ${_gnupg_decrypt_opts} | tar -xf -
cd "${_old_pwd}"
```

If you wish to see what other features are working and tested in a publicly verifiable way, then check out the Travis-CI build logs (especially near the end of the logs) you'll find there's some other fancy things being worked on in relation to encryption and decryption of nearly any data.

Share Improve this answer Follow

edited Nov 22, 2016 at 16:44
Toby Speight
5,146 1 28 38

answered Nov 21, 2016 at 20:56
S0AndS0
133 1 1 6

0

Use FinalCrypt - Unbreakable One-Time Pad OpenSource File / Directory Encryption (GUI & CLI)

It creates One-Time Pad keys by itself

```
java -cp FinalCrypt.jar rdj/CLUI --encrypt --password-prompt -k My-Key-Directory/ -t
My-Test-Directory/
```

Password:

Started encrypting 4 files totally 249,7 MiB

🔒 "/home/ron/My-Test-Directory/Video/Eerebegraafplaats.mp4.bit" 🗝 ✔ 📝 ✔ 🔒✔ ₵✔ 🗑✔ SHA-256: "C1E3F3A3545FEA026F3FB344F3D0798B54820B7F9AD9AAC4BE9FD1E955F947DA"->"D53FCEADDF542AC3655B547778911F786C2C2BDD327E0618A9E7F77B57792DEA" 58,4% 🔒 "/home/ron/My-Test-Directory/Video/castle-waxjo-sweden.mp4.bit" 🗝 ✔ 📝 ✔ 🔒✔ ₵✔ 🗑✔ SHA-256: "8AEFC9744143451F32B82BBAC6A4291BC76C747A6DA1EA024702AA51A966F810"->"323618B7ED12A1F92D8FFB306CEEC6DFFED6862B7BF3922902E8AED29DF57ECE" 91,2% 🔒 "/home/ron/My-Test-Directory/Brother_HL-2170W-usaeng_quick-setup.pdf.bit" 🗝 ✔ 📝 ✔ 🔒✔ ₵✔ 🗑✔ SHA-256: "0858D2D5A8CF118D40B517CD4A1F8D31D9F5A21221F75BD764B5E363FC1431FE"->"266CE42027F891DECF109D7A9DD69E8B42C0E43D35E952BEB89F7C7EA2DBE92C" 95,7% 🔒 "/home/ron/My-Test-Directory/Brother dsmobile 700d_uke_usr.pdf.bit" 🗝 ✔ 📝 ✔ 🔒✔ ₵✔ 🗑✔ SHA-256: "8D718D2F29EF05BEB347D6920B3BFF5269685421B428E8D3ADFF569F67A716E0"-

> >"88A98D893B6D1E540039D3E9BC0B0C19B46A10A209967F3235D5DEEBF073EC1E" 100,0%
>
> Finished encrypting [4 / 4] files totally [249,7 MiB / 249,7 MiB] in 7,3 seconds (average: 34,2 MiB/s)

```
java -cp FinalCrypt.jar rdj/CLUI --decrypt --password-prompt -k My-Key-Directory/ -t
My-Test-Directory/
```

> Password:
>
> Started decrypting 4 files totally 124,9 MiB
>
> 🔓 "/home/ron/My-Test-Directory/Video/castle-waxjo-sweden.mp4" 🖃 ✔ 🔓✔ ₡✔ 🗑✔ SHA-256: "323618B7ED12A1F92D8FFB306CEEC6DFFED6862B7BF3922902E8AED29DF57ECE"->"8AEFC9744143451F32B82BBAC6A4291BC76C747A6DA1EA024702AA51A966F810" 32,8% 🔓 "/home/ron/My-Test-Directory/Video/Eerebegraafplaats.mp4" 🖃 ✔ 🔓✔ ₡✔ 🗑✔ SHA-256: "D53FCEADDF542AC3655B547778911F786C2C2BDD327E0618A9E7F77B57792DEA"->"C1E3F3A3545FEA026F3FB344F3D0798B54820B7F9AD9AAC4BE9FD1E955F947DA" 91,2% 🔓 "/home/ron/My-Test-Directory/Brother dsmobile 700d_uke_usr.pdf" 🖃 ✔ 🔓✔ ₡✔ 🗑✔ SHA-256: "88A98D893B6D1E540039D3E9BC0B0C19B46A10A209967F3235D5DEEBF073EC1E"->"8D718D2F29EF05BEB347D6920B3BFF5269685421B428E8D3ADFF569F67A716E0" 95,5% 🔓 "/home/ron/My-Test-Directory/Brother_HL-2170W-usaeng_quick-setup.pdf" 🖃 ✔ 🔓✔ ₡✔ 🗑✔ SHA-256: "266CE42027F891DECF109D7A9DD69E8B42C0E43D35E952BEB89F7C7EA2DBE92C"->"0858D2D5A8CF118D40B517CD4A1F8D31D9F5A21221F75BD764B5E363FC1431FE" 100,0%
>
> Finished decrypting [4 / 4] files totally [124,9 MiB / 124,9 MiB] in 3,4 seconds (average: 36,3 MiB/s)

It also has a GUI

Just trying to help the community...

Share Improve this answer Follow

answered Jul 14, 2019 at 7:23

Ron de Jong

1 1

See Comments on FINALCRYPT. – Scott - Слава Україні Jul 14, 2019 at 7:44

FinalCrypt 5 added Auto Key and creates OTP Keys automatically so the discussion Scott is referring to is no longer relevant – Ron de Jong Jul 14, 2019 at 10:38

I would like to see a description of how it works that's more technical and less hand-waving.   The issue is that one-time pads (1) are great for *transmission* of data, and lousy for *storage,* and (2) should be *random.*   If FinalCrypt's OTPs are truly random, then they must be *stored,* which compromises security.   If they can be regenerated, then they are not random, but only pseudo-random, and so they are not proper OTPs.   … (Cont'd) – Scott - Слава Україні Jul 14, 2019 at 15:18

(Cont'd) …   Their page on Auto Key Management indicates that the OTPs are stored "on a detachable external (USB) drive. "   OK, that could work.   But, if you have to attach your USB drive every time you want to decrypt your file (and given that an OTP must be at least as big as the file it encrypts), you might as well just ***store your files on the removable drive*** and not bother with encryption. … (Cont'd) – Scott - Слава Україні Jul 14, 2019 at 15:18

(Cont'd) …   Also, the main FinalCrypt page says "most crypto software uses broken AES …", but claims that AES is "broken" seem to be greatly exaggerated.   Related: Why is AES considered to be secure? (on Cryptography Stack Exchange). – Scott - Слава Україні Jul 14, 2019 at 15:18

0

У меня нет всех ваших требований, но если вы хотите прозрачно зашифровать данные, хранящиеся на диске, обратите внимание на fscrypt.

Это инструмент, реализованный в ядре и в командной строке , который позволяет шифровать каталоги.

на данный момент вам необходимо использовать ext4 в качестве файловой системы.

Вот шаги, которые нужно сделать:

1. включите шифрование на устройстве файловой системы: `tune2fs -O encrypt /dev/nvme0n1p3`
2. Boostrap fscrypt на файловой системе, смонтированной на устройстве: `fscrypt setup`
3. Зашифровать каталог в файловой системе: `fscrypt encrypt /opt/encrypted/`

Теперь все файлы, которые вы поместите внутрь, будут зашифрованы.

Делиться Улучшить этот ответ Следовать

ответил 15 апр. 2021 г. в 8:17
Батист Милле-Матиас
101 4