How to encrypt a file with private key

Asked 5 years, 11 months ago Modified 8 months ago Viewed 28k times

3

I want to encrypt a file with a private key and decrypt it with a public key. A public key will be embedded in my app. So I want to have a guarantee that the file was created by me. How can I use gpg or openssl to implement it.

openssl gpg signature

4

Share Improve this question

Follow

edited Jul 19, 2016 at 0:13

Gilles 'SO- stop being evil'

752k 180 1561 2052

asked Jul 18, 2016 at 19:17

NoTrust

143 1 1 7

stackoverflow.com/questions/2295786/… security.stackexchange.com/questions/86595/… – frostschutz Jul 18, 2016 at 19:28

4 What you're describing sounds like signing (making sure the file was created by somebody who knows the private key), not encrypting (making sure only somebody with the private key can read it)? – Ulrich Schwarz Jul 18, 2016 at 19:52

PHP also has a function for encrypting with a private key. I have not used it myself, but did want to share it here. php.net/manual/en/function.openssl-private-encrypt.php – Jonathan Sep 30, 2021 at 0:33

4 Answers

Sorted by: Highest score (default)

8

Use OpenSSL to do that. Follow a simple example:

To encrypt a file:

```
openssl rsautl -encrypt -inkey public_key.pem -pubin -in <decrypted file> -out
<encrypted file>
```

To decrypt a file:

```
openssl rsautl -decrypt -inkey private_key.pem -in <encrypted file> -out
<decrypted file>
```

Share Improve this answer

Follow

edited Jul 24, 2018 at 14:23

Marco

893 9 19

answered Jul 18, 2016 at 20:09

Franciscon Santos

535 3 7

6 He is asking how to encrypt using Private Key. – Ibrar Ahmed Sep 21, 2018 at 18:52

6

You can encrypt with a private key and decrypt with its public key:

- To encrypt

```
$ TEXT="proof that private key can encrypt and public key can decrypt"
$ echo "$TEXT" | openssl rsautl -sign -inkey private.key -in - -out - |
base64 > encrypted.txt
```

- To decrypt

```
$ cat encrypted.txt | base64 -d | openssl rsautl -verify -pubin -inkey
public.key -in -
proof that private key can encrypt and public key can decrypt
```

As you can see, the decrypted file correctly matches the text we wrote into it in the encryption step.

Share Improve this answer

Follow

edited Aug 10, 2020 at 15:39

AdminBee

18.5k 16 41 64

answered Aug 10, 2020 at 14:42

rsmoorthy

161 1 2

This is the right answer. Thank you very much. – Dan Ortega Oct 11, 2020 at 4:38

1 As explained in other answers, signing is NOT the same as encryption. For one thing, a signature is a fixed length. Your example 61 bytes only works because it is a very small piece of data you are using. Try making your text longer and it will fail with: data too large for key size – Phil_1984_ Oct 24, 2021 at 1:15

5

It makes no sense to encrypt a file with a private key.

Using a private key to attach a tag to a file that guarantees that the file was provided by the holder of the private key is called **signing**, and the tag is called a **signature**.

There is one popular cryptosystem (textbook RSA) where a simplified (insecure) algorithm uses has public and private keys of the same type, and decryption is identical to signature and encryption is identical to verification. This is not the case in general: even RSA uses different mechanisms for decryption and signature (resp. encryption and verification) with proper, secure padding modes; and many other algorithms have private and public keys that aren't even the same kind of mathematical objects.

So you want to sign the file. The de facto standard tool for this is GnuPG.

To sign a file with your secret key:

```
gpg -s /path/to/file
```

Use the `--local-user` option to select a secret key if you have several (e.g. your app key vs your personal key).

Transfer `file.gpg` to the place where you want to use the file. Transfer the public key as well (presumably inside the application bundle). To extract the original text and verify the signature, run

```
gpg file.gpg
```

If it's more convenient, you can transfer `file` itself, and produce a separate signature file which is called a detached signature. To produce the detached signature:

```
gpg -b /path/to/file
```

To verify:

```
gpg file.gpg file
```

You can additionally encrypt the file with the `-e` option. Of course this means that you need a separate key pair, where the recipient (specified with the `-r` option) has the private key and the producer has the public key.

Share Improve this answer

Follow

edited Apr 13, 2017 at 12:48

Community Bot

1

answered Jul 19, 2016 at 0:13

Gilles 'SO- stop being evil'

752k 180 1561 2052

2 *It makes no sense...* Disagree. Consider deployment of a secure package to multiple endpoints from a *guaranteed* single source. The encrypted package provides both content security and, intrinsically, the signature. Alternately, the package is encrypted with a symmetric key which is protected asymmetrically then bundled together for delivery. A wholly, asymmetrically encrypted package is preferable -- even if not possible today. – bvj Apr 1, 2019 at 19:36

@bvj Then you aren't encrypting with a private key, as in, the private part of a key pair for an asymmetric cryptographic scheme. You're encrypting with a *secret key*. (Terminology isn't completely standardized, but most of the world uses “private key” only in the context of public-key cryptography, and uses “secret key” in the context of symmetric cryptography.) With public-key cryptography, if you're using a private key, you're either decrypting or signing, not encrypting. – Gilles 'SO- stop being evil' Apr 1, 2019 at 19:56

Thanks, @Gilles. Notwithstanding convention, isn't it possible via RSA to encrypt with the *private* key and subsequently decrypt with only the *public* key? The endpoints I referred to would be securely configured with the *public* key. Thereafter, they'd receive payloads that could certainly be signed with the sender's *private* key, but preferably receive an additional symmetric key encrypted by the sender's *private* key. The symmetric key could then decrypt the protected payload. Signing is essential, obscurity is preferable. The endpoints could be IoTs for example. – bvj Apr 1, 2019 at 20:35

2 @bvj No: you encrypt the symmetric key with the recipient's public key. Even if you could encrypt with the sender's private key, anybody could decrypt it with the corresponding public key, so that would make the encryption pointless. Encrypting with a private key, or signing with a public key, is technically possible with RSA (but typically not with other public-key algorithms, as in, you can perform the mathematical operation, but it doesn't have any interesting security property. – Gilles 'SO- stop being evil' Apr 1, 2019 at 20:52

So basically two sets of key pairs are required where both the sender and recipient each have their own private key. Thank you, Gilles! – bvj Apr 1, 2019 at 21:14

0

You're speaking of an app. That means you probably want to use a library, not a command line. How to do that is out of scope for UNIX & Linux SE (and you didn't even tell us what language your app is written in), but there are several libraries which can do what you want:

- OpenSSL's libcrypto has several primitives to deal with signature verification
- GPGME is a library written by the author's of GnuPG, which allows many things, among them signature verification.
- GnuTLS also exposes its internal cryptographic primitives, however they don't recommend it.

I'm sure there are more, these are just the most popular solutions. Most of these libraries have bindings for other languages, too, in case you're not using C.

Share Improve this answer Follow

answered Jul 19, 2016 at 6:50

Wouter Verhelst

8,755 17 42