Security

How Do I Verify a PGP Signature?

9 months ago • by David Adams

PGP (Pretty Good Privacy) is a public key-based cryptography program. PGP complements symmetric-key with asymmetric-key algorithms, making this software a hybrid cryptographic system, often called ***hybrid cryptosystem***.

PGP is not only used to secure information from cyber threats but also to check file integrity.

This tutorial explains easily how PGP works and how to verify PGP signatures.

How PGP Works

The image below depicts a PGP public key. This PGP public key can be decrypted only with a specific private PGP key. The issuer of the public key below also issued a private PGP key since they are generated in the same process. He only shares the public key.
If you take his public key to encrypt a message to him, he will be able to decrypt the message using his private key. Only his private key can decrypt the message you encrypted using his public key.

-----BEGIN PGP PUBLIC KEY BLOCK-----

xo0EYKxFvwEEAPL+RsqSC8WkpMUAFUArYLp+oPWYJz5cu+IXwSgEXaaun/hwppQW
EkEuvqasKtL8qrv5GYu3fxvQhPMJSwUFgkgLygHt/hj5UstW4Vn7aXgcwrVrDR/j
4mWbMkFLjfmFsSX8r5DpExCEx4KJc6YVPpM8o4QLbrY/jTMZ3xKB9qYfABEBAAHN
JExJbnV4IEhpbnQgPGxpbnV4aGludEBsaW51eGhpbnQuY29tPsKtBBMBCgAXBQJg
rEW/AhsvAwsJBwMVCggCHgECF4AACgkQc134Z7D55f+ouAP9HOe1NsLWh/aEvnsO
IxjlRowZGF0kNiMjB+U/t8Lfrwfv7Hu5LcAV/K+Ou//KvS9j063/dNyYngZaQRaS
bd9OjOroTFcJl2KiY1Auuq78pe24cdoGgfw2MO5VKFfqmvOaQ/ifuESwcSm/M4nR
y0ctclzg0gSKlbbcTUGcpWlYJWPOjQRgrEW/AQQAtSYw/+9QT6kOz4So/H00dQr9
mQpKClA/o1gN9xnI9hDSL+UBr3bsE4xcXut3i90M3P5wcuZsAYDis5ZXJ6Z6ARPv
Zi9amd0iaga0TAYUUmP+5kY9HSgKHYR/9dix+sfMHonYteqk+S2FSqp05EUhqA7y
B4zgATpfN8iCTDv5rPsAEQEAAcLAgwQYAQoADwUCYKxFvwUJDwmcAAIbLgCoCRBz
XfhnsPnl/50gBBkBCgAGBQJgrEW/AAoJEPgQmm5Eu9s4kpMD/jr9+Xs+IXtseaPR
Al0l+3P7EAsF8UtN1qMZy150VDolmXE1CmOA/k/azCQ13EiYBPkCUJrHZ7lgruBx
4S6nFkbXKjdxKz/2LWm/q33fZWnT8rQKLaTjVZ6sYaZMHSPWsRbAc81CeDVyzXY8
7w8O10THPV3eo7GUu+MiqILXTA98evlEAKaGGkoPq4W788zPAcWjXV3iNnkmWuj7
a6b/TCxcYs8OCBA5TVS7Mafm/621ilRH02crJmRAPVr+7rMadTQ7iijW1mczmNJM
avCpBJryluLoAP/q2ORfGhyR1biwCeDWOJWA6uYh0ZkyFXwTwviLDLZgh+SJVEFl
J2JRMz6gcXVAzo0EYKxFvwEEANVxjpSAV3kU+QTU6amDM+PKid0atMOVRrl45GrL
b717NM9h66jQKBI0H90FZ9tnnnxW9mLzKwygUYucVoz1sQP60tYGEsVb3Cho7zOE
NEVvEeSXKltEF3SkUfhIezy350g87qSxBci+sP8Gj4bg0iCw7n2F6WGfUF/t5YvE
QOkBABEBAAHCwIMEGAEKAA8FAmCsRb8FCQ8JnAACGy4AqAkQc134Z7D55f+dIAQ
AQoABgUCYKxFvwAKCRBAMe0N2o6FPc/GA/9PV1Bk5VPPbxKKG3+c2sA7rwzCEKGC
8luMruZFdlhFxEf2FFNQX8BHoBN1mkUAzg7e0L8tYi7li6j6Mwru2H/ND0lVauzh
oaojEXG8FEVDfKrXfFuk8xcEK8gewBYOYAHoA5dvkn1leqTDTND0WAgnI24KKiEx
qOrUhw7VYxxXh7fFA/9/LGpFOqEY1KtFeIZokmo5orTNTxSiYXB+bgrHbVlBnieX
pk/okqgQAHxO/NoBkAXsMY+H4YZoCnJwefeZ83xfoRssyvHadqkI/0TVeB2WA/j3
uhGYlEhliLfZcMfQPqdCl6OmXt17gLesbJJNam+fOVEnGJnydjlcZfrNQsPWrw==
=4aLt
-----END PGP PUBLIC KEY BLOCK-----

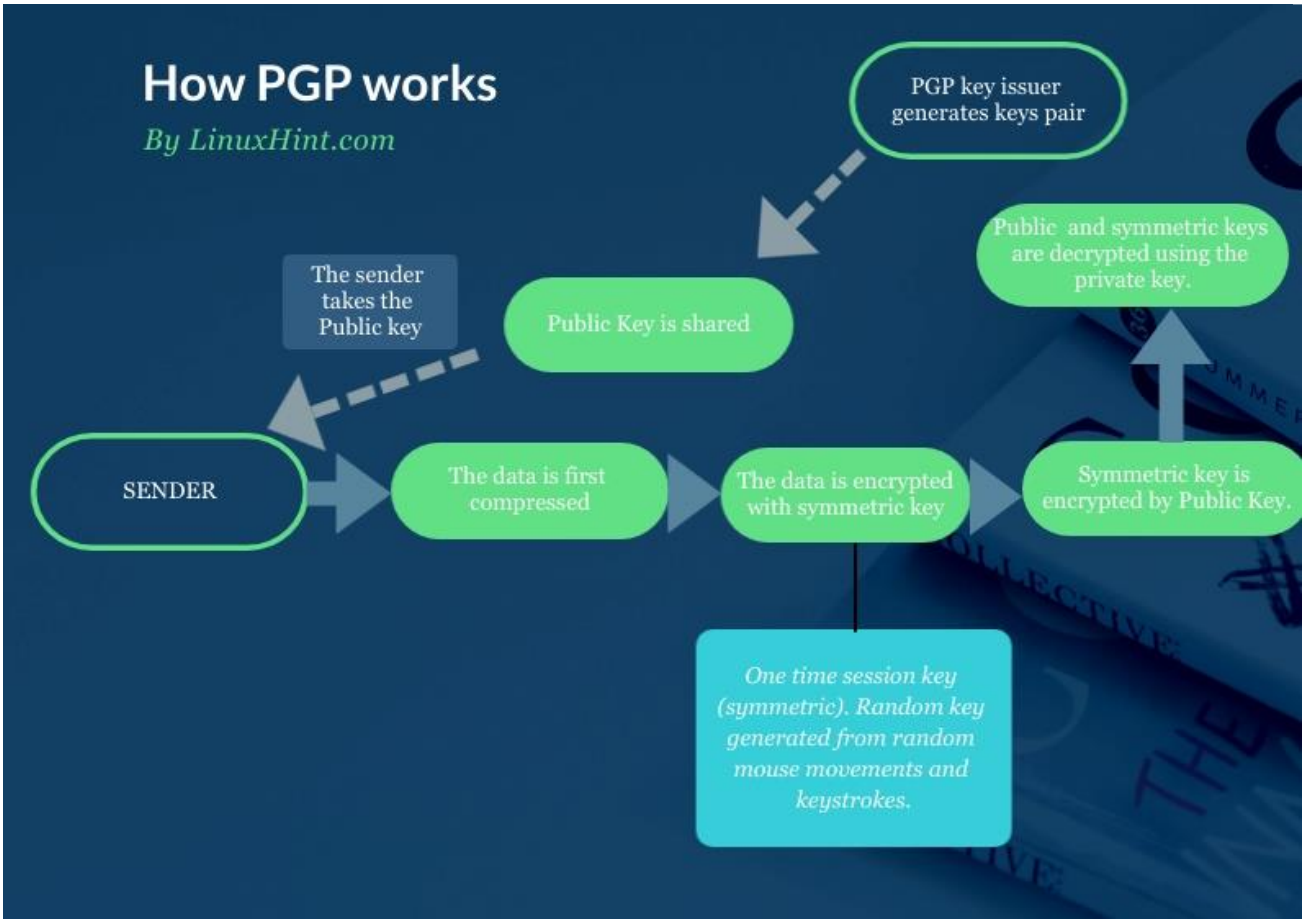
The information is encrypted using the public key, and decrypted using the private key. This is called ***asymmetric encryption***.

So even if an attacker manages to intercept the message without the private key, he is unable to see the message content.

The advantage of asymmetric encryption is the simplicity to exchange keys. But its disadvantage is it can't encrypt large amounts of data, and that's why PGP implements both of them.

Symmetric encryption is applied when the public key is used to encrypt the protected data. With the public key, the sender does two things: first generates the symmetric encryption to protect the data, and then it applies asymmetric encryption, which does not encrypt the data itself, but the symmetric key, which protects the data.

To be more technical, before the symmetric key is applied, the data is also compressed before being encrypted with the symmetric key and public key. The following chart flow shows the whole process:



PGP Signatures

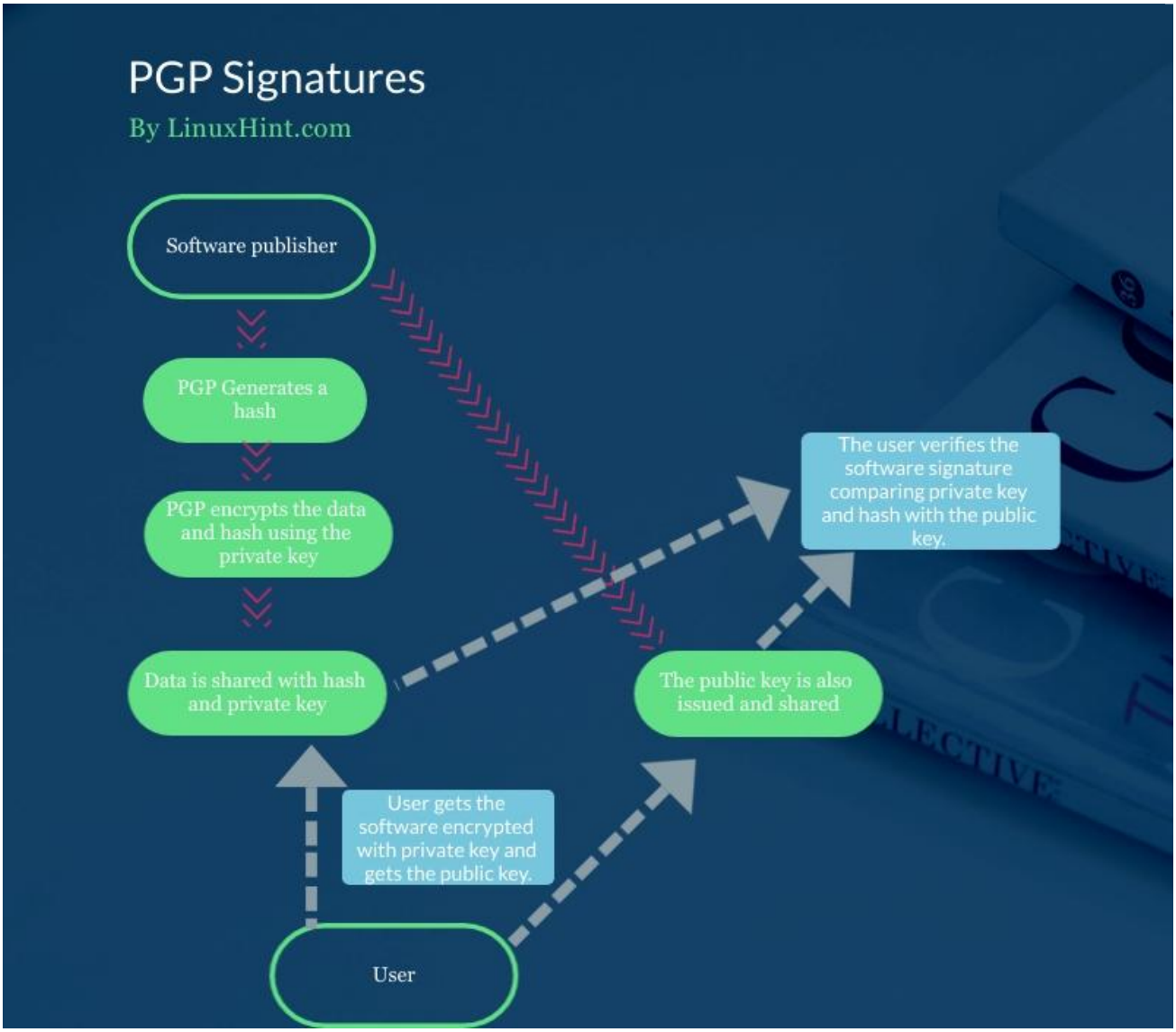
PGP is also used to check packages' integrity. This is achieved through digital signature, which can be done with PGP.

First, PGP generates a hash that is encrypted with the private key. Both private key and hash can be decrypted using the public key.

PGP creates a digital signature, for example, for an ISO image using DSA or RSA algorithms. In this case, the private key is attached to the software or ISO Image, contrary to the operation described previously. The public key is also shared.

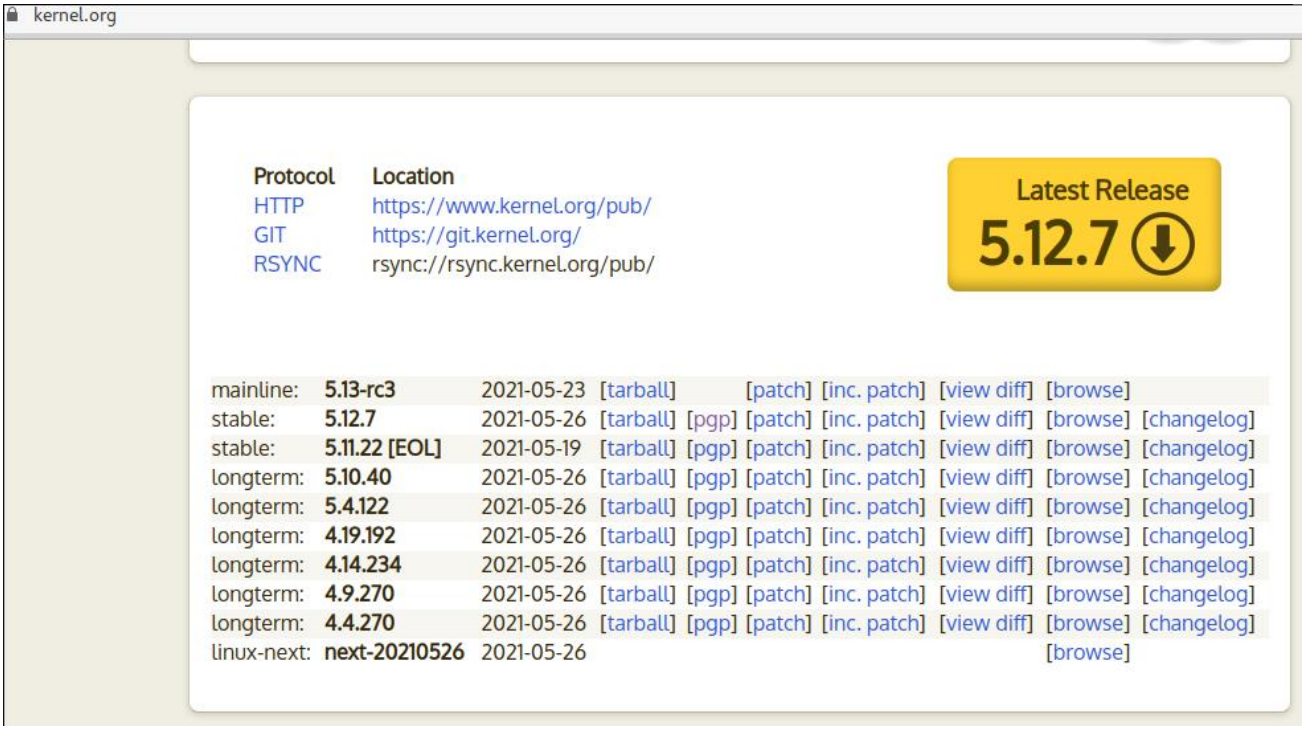
Users use the public key to verify the signature attached to the released software.

The following chart flow shows how the private key and hash is attached to the software and how the user takes the software with the attached hash and private key together with the public key to verify the signature:



How Do I Verify a PGP Signature?

The first example shows how to verify the Linux kernel signature. To try it, access https://kernel.org and download a kernel version and its PGP file. For this example, I will download files **linux-5.12.7.tar.xz** and **linux-5.12.7.tar.sign**.



The first example shows how to verify the signature with a single command. According to the man page, this option combination is going to be deprecated in future versions. However, it is still widely used, and while the specific combination will be deprecated, the options will remain.

The first option **–keyserver-options** allows defining options for the keyserver where public keys are stored. Basically, this allows the implementation of public keys fetching options.

The **–keyserver-options** is combined with the **–auto-key-retrieve** option to automatically retrieve public keys from a keyserver when verifying signatures.

To find the public keys, this command will read the signature looking for a defined preferred keyserver or signer’s ID through a lookup process using Web Key Directory.

```
gpg --keyserver-options auto-key-retrieve --verify linux-5.12.7.tar.sign
```

As you can see, the signature is good, but there is a warning message saying gpg can't confirm the signature belongs to the owner. Anyone can issue a public signature as Greg Krohan-Hartman. You know the signature is legitimate because you trust the server you have downloaded it from. In this case, it is specified in the .sign downloaded from kernel.org. This warning is always present, and you can avoid it by adding signatures to a signature trusted list using the option *–edit-key trust*. The truth is no user does it, and the Gpg community requested the warning removal.

Verifying SHA256SUMS.gpg

In the following example, I will verify the integrity of an old Kali Linux image I found in my box. For this purpose, I downloaded the SHA256SUMS.gpg and SHA256SUMS files belonging to the same iso image.

Once you download an iso image, the SHA256SUMS.gpg, and SHA256SUMS, you need to get the public keys. In the following example, I fetch the keys using **wget** and **gpg –import** (Kali verification instructions link to this key server).

Then I verify the file integrity by calling gpg with the **–verify** argument:

```
wget -q -O - https://archive.kali.org/archive-key.asc | gpg --import
gpg --verify SHA256SUMS.gpg SHA256SUMS
```

As you can see, the signature is good, and the verification was successful.

The following example shows how to verify a NodeJS download. The first command returns an error because there is no public key. The error indicates I need to search for the key 74F12602B6F1C4E913FAA37AD3A89613643B6201. Usually, you can also find the key ID in the instructions.

By using the option **–keyserver**, I can specify the server to search for the key. By using the option **–recv-keys**, I retrieve keys. Then the verification works:

```
gpg --verify SHASUMS256.txt.asc
```

I copy the key I need to fetch, and then I run:

```
gpg --keyserver pool.sks-keyservers.net --recv-keys
74F12602B6F1C4E913FAA37AD3A89613643B6201

gpg --verify SHASUMS256.txt.asc
```

Searching gpg Keys:

If auto retrieving keys doesn't work and you can't find the verification-specific instructions, you can search the key in a keyserver using the option **–search-key**.

```
gpg --search-key 74F12602B6F1C4E913FAA37AD3A89613643B6201
```

As you can see, the key was found. You can also retrieve it by pressing the number of the key you want to retrieve.

Conclusion

Verifying downloads' integrity may prevent serious problems or explain them, for example, when downloaded software doesn't work correctly. The process with gpg is pretty easy, as shown above, as long as the user gets all the necessary files.

Understanding asymmetric encryption or public and private keys-based encryption is a basic need to interact safely on the internet, for example, using digital signatures.

I hope this tutorial on PGP signatures was helpful. Keep following Linux Hint for more Linux tips and tutorials.

ABOUT THE AUTHOR

David Adams

David Adams is a System Admin and writer that is focused on open source technologies, security software, and computer systems.

View all posts

RELATED LINUX HINT POSTS

- Protecting your files and folders
- Basic Pfsense Configuration Tutorial
- Encryption vs. Hashing
- Types of Cryptography
- Cryptographic Hash Functions
- VLAN Hopping Attack and Mitigation
- History of Cryptography