

APT - Keyrings

APT repository signing uses GPG keys to authenticate packages and repository metadata before installation. Modern Debian and Ubuntu systems prefer isolated, unarmored or armored keyring files coupled directly to individual .list or .sources configuration files rather than global legacy keyrings.

File/Directory Path	Status	Description
/etc/apt/keyrings/	Recommended	Standard directory for storing third-party repository GPG keys. Kept isolated per repository and explicitly referenced via the signed-by option in APT sources configuration files.
/usr/share/keyrings/	Standard	Directory managed by OS package manager (dpkg/apt) containing official system distribution signing keys (e.g., ubuntu-archive-keyring.gpg or debian-archive-keyring.gpg).
/etc/apt/trusted.gpg.d/	Legacy / Active	Directory containing fragment keyrings (or export files with extension .gpg or .asc). APT trusts all keys in this directory globally for any repository unless restricted.
/etc/apt/trusted.gpg	Deprecated	Legacy primary keyring file where apt-key add previously stored GPG keys. Deprecated because any key added here is trusted globally across every repository.
/etc/apt/sources.list	Active	Main system configuration file for APT software repositories. Accepts signed-by=/path/to/key.gpg directives inline to pin a key to a specific repository line.
/etc/apt/sources.list.d/	Active	Directory containing modular repository source configuration files (.list or .sources). Frequently pairs with custom keyrings stored under /etc/apt/keyrings/.
Repository server: InRelease	Remote File	Combined repository index file containing package hashes and an inline GPG signature created by the repository maintainers.
Repository server: Release	Remote File	Repository metadata index listing package file hashes. Used alongside Release.gpg for verification.
Repository server: Release.gpg	Remote File	Detached GPG signature file used to authenticate an unsigned Release file.

- Storing repository signing keys in global keyrings creates a serious security risk because any key added to /etc/apt/trusted.gpg or /etc/apt/trusted.gpg.d/ can authenticate packages from any repository on the system, not just the one it was intended for.
- Modern APT security requires storing keys as individual files in /etc/apt/keyrings and explicitly linking them to their repositories using the signed-by option, ensuring each key can only authenticate its designated repository.
- ASCII-armored PGP keys must be converted to binary GPG keyring format using `gpg --dearmor` before APT can use them, as APT only accepts keys in binary .gpg format for repository authentication.
- Every modern repository configuration must include the signed-by=/etc/apt/keyrings/<keyring>.gpg parameter in its definition, otherwise APT cannot determine which key to use and will fail to authenticate the repository.
- Repository entries can be managed in three equivalent ways: adding lines directly to /etc/apt/sources.list, creating separate .list files in /etc/apt/sources.list.d/, or using the structured deb822 format in .sources files.

- Understanding directory purposes is critical: `/etc/apt/keyrings` stores administrator-managed keys, `/usr/share/keyrings` contains package-managed keys, and `/etc/apt/trusted.gpg.d/` represents the deprecated legacy approach that should be avoided.
- While keys can be retrieved from key servers using `gpg --recv-keys`, downloading them directly via HTTPS is more reliable and avoids potential issues with slow, unreliable, or blocked key servers, though both methods work with scoped keyrings.

Step 1 — Identifying the Components and Key Format

GPG, or GNU Privacy Guard, is an open-source encryption program used for signing, encrypting, and decrypting files and directories. GPG files are usually keyrings, which are files that hold multiple keys. `gpg` is GPG's command-line tool that can be used to authorize external repositories for use with `apt`. However, `apt` expects keys to be in a GPG keyring format (binary `.gpg` files). In order to use this command-line tool with ASCII-armored PGP files, you must convert them into this format. PGP, or Pretty Good Privacy, is a proprietary alternative application.

Projects that require adding repositories with key verification will always provide you with a public key and a repository URI representing its exact location. For our Elasticsearch example, the documentation gives these components on their installation page.

Here are the components given for Elasticsearch:

```
Key: https://artifacts.elastic.co/GPG-KEY-elasticsearch
Repository: https://artifacts.elastic.co/packages/9.x/apt stable main
```

Step 2 — Downloading the Key and Converting to an apt Compatible File Type

On modern Debian systems, the `/etc/apt/keyrings` directory is the recommended location for storing administrator-managed keyrings. Create this directory if it does not already exist:

```
# mkdir -p /etc/apt/keyrings
```

Then download, check and convert the key:

```
# curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch > tmp.key
```

Next, you have to determine whether you are given a PGP or GPG file to work with:

```
# file tmp.key
-----
tmp.key: PGP public key block Public-Key (old)

# file tmp.key
-----
tmp.key: OpenPGP Public Key Version 4, Created Mon Sep 16 15:07:54 2013, RSA
(Encrypt or Sign, 2048 bits); User ID; Signature; OpenPGP Certificate
```

The first result indicates that this is actually a PGP key file. Since apt expects keys to be stored in a GPG keyring format, convert the to gpg format with following:

```
# gpg --dearmor -o /etc/apt/keyrings/elastic-9.x.gpg tmp.key
```

Step 3 — Adding the Repository to Your List of Package Sources

Next step is adding the repository to the apt package sources while explicitly linking it to the key you obtained. There are three methods to achieve this. Apt pulls sources from a central sources.list file, .list files in the sources.list.d directory, and .sources files in the sources.list.d directory. Though there is no functional difference between these options, using separate files in sources.list.d is generally easier to manage and maintain.

Option 1 — Adding to sources.list Directly

This involves inserting a line describing the source directly into /etc/apt/sources.list, the primary file containing apt sources. There are multiple sources in this file, including the default sources that come with Debian. It is perfectly acceptable to edit this file directly, though Option 2 and Option 3 present more modular solutions that are easier to edit and maintain. Add the external repository to the bottom of the file:

```
# nano /etc/apt/sources.list
-----
deb [arch=amd64,arm64 signed-by=/etc/apt/keyrings/elastic-9.x.gpg]
https://artifacts.elastic.co/packages/9.x/apt stable main
```

This line contains the following information about the source:

- deb: Specifies that the source uses a Debian package format (Regular Binary) or deb-src (Source), depending on if you want a package or the source of the package.
- arch=amd64,arm64: Specifies the architectures the APT data will be downloaded for
- signed-by=/etc/apt/keyrings/elastic-9.x.gpg: Specifies the key used to authorize this source. This is a required addition in modern APT usage and ensures that the key is only trusted for this repository
- <https://artifacts.elastic.co/packages/9.x/apt> stable main: The URI representing the repository location

Note: signed-by is **optional**, not mandatory. When a deb line has no signed-by option, APT falls back to checking the repo's release signature against all keys in its default trust stores: /etc/apt/trusted.gpg or in every file in /etc/apt/trusted.gpg.d/*.gpg.

Option 2 — Creating a New .list File in sources.list.d

Create a new file in the sources.list.d directory. Apt parses both this directory and sources.list for repository additions. Create a new file and insert the appropriate line. The file is named elastic-9.x.list

in the following example, but any unique filename will work:

```
# echo "deb [arch=amd64,arm64 signed-by=/etc/apt/keyrings/elastic-9.x.gpg]
https://artifacts.elastic.co/packages/9.x/apt stable main" | sudo tee
/etc/apt/sources.list.d/elastic-9.x.list > /dev/null
```

Option 3 — Creating a .sources File in sources.list.d

The third method writes to a .sources file instead of a .list file. This method uses the **deb822 multiline format**, which is more structured and less ambiguous than the single-line deb format, while remaining functionally equivalent. Create a new file:

```
# nano /etc/apt/sources.list.d/elastic-9.x.sources
-----
Add the external repository using the deb822 format:

Types: deb
Architectures: amd64 arm64
Signed-By: /etc/apt/keyrings/elastic-9.x.gpg
URIs: https://artifacts.elastic.co/packages/9.x/apt
Suites: stable
Components: main
```

This format organizes the same information as the one-line format, but in a clearer, field-based structure. One difference is that it uses spaces instead of commas when specifying multiple values (for example, amd64 arm64 instead of amd64,arm64).

From:
<https://wiki.oscardegroot.nl/> - HomeWiki

Permanent link:
<https://wiki.oscardegroot.nl/doku.php?id=linux:debian:apt-keyrings&rev=1788672104>

Last update: 2026/09/06 05:21

