# TOTP authentication for sudo

PAM stack design, password plus TOTP, caching and recovery

## How sudo authentication is assembled

sudo consults its policy (usually sudoers) to determine whether a command is permitted and whether authentication is required. When authentication is needed, it uses PAM. On many Ubuntu installations /etc/pam.d/sudo includes common-auth for password verification; additional PAM modules can require a TOTP code.

PAM configuration is order-sensitive and distribution-specific. A module declared auth required normally must succeed, although later modules may still run. A configuration that bypasses password authentication or accidentally demands an extra password can produce very different behaviour from the intended password-plus-TOTP design.

```
sudo -l
sudo grep -nE "^(auth|@include)" /etc/pam.d/sudo
```

## Installing and enrolling an authenticator

On Ubuntu, libpam-google-authenticator supplies pam_google_authenticator.so and the google-authenticator enrolment utility. The name of the module does not require use of Google's phone application: a compatible TOTP authenticator can scan the same enrolment QR code.

Run the enrolment utility as the account that will authenticate, so its secret file belongs to that user. Preserve recovery codes securely. Avoid displaying live secrets in logs, terminal recordings or shared documentation.

```
sudo apt install libpam-google-authenticator
google-authenticator
ls -l ~/.google_authenticator
```

## Adding the second check

For a standard Ubuntu sudo PAM stack, a common arrangement adds the Google Authenticator module as a required authentication check while retaining the existing common-auth password check. The precise insertion point and behaviour should be validated against the installed PAM file. Some environments use additional sudo service files such as sudo-i; check which service each invocation uses.

Before editing, retain an already-authorised session and ensure console recovery. Do not replace the entire PAM file with a minimal example; preserve account and session rules supplied by the distribution.

```
sudo cp -a /etc/pam.d/sudo /etc/pam.d/sudo.before-totp
# Typical additional auth rule in /etc/pam.d/sudo:
auth required pam_google_authenticator.so
# Retain the existing @include common-auth rule.
```

## Why a prompt may not appear

sudo caches successful authentication for a configurable period. A second sudo command may not ask for either password or TOTP until the timestamp expires. sudo -k invalidates the cached credentials for testing. NOPASSWD rules, root invocation and other sudoers settings can also suppress PAM authentication.

Some distributions distinguish sudo and sudo-i PAM services. Check /etc/pam.d/sudo-i if present and confirm that the intended rules apply to sudo -i. Do not assume every sudo invocation follows the same PAM stack.

```
sudo -k
sudo id
sudo -k
sudo -i
ls -l /etc/pam.d/sudo*
```

## Troubleshooting and rollback

If authentication fails, examine PAM journal messages, file permissions, account enrolment and time synchronisation. A secret file owned by the wrong account or inaccessible under PAM's privilege context can cause failure. A lost phone should be handled through a defined recovery process, not by weakening authentication for everyone.

Changes to PAM can lock out administrative access. Keep an alternate privileged session and verify both a fresh sudo command and sudo -i before considering the change complete.

```
timedatectl status
sudo journalctl --since today | grep -Ei "sudo|pam_google_authenticator"
sudo -k
```

Reference scope: commands and defaults vary by distribution and software version. Consult the locally installed manual pages for exact syntax and behaviour.