git-crypt Transparent repo encryption

CONCEPTS

> Files are encrypted only when committed; your working tree stays plain text while the repo is unlocked.

Working tree plain text
↓ `git add (clean filter)`
Repository encrypted blobs
↓ `git push`
Remote ciphertext only

> Rules must exist before the files are added; earlier commits stay readable in history.

SETUP

`git-crypt init`
Generate a key and prepare the repo

`git-crypt help [<command>]`
Show help for a command

`git-crypt version`
Print the installed version

CHOOSING FILES

`<path> filter=git-crypt diff=git-crypt`
One .gitattributes rule: encrypt a file

`secrets/** filter=git-crypt diff=git-crypt`
Encrypt a whole directory tree

`*.key filter=git-crypt diff=git-crypt`
Encrypt by pattern

`.gitattributes !filter !diff`
Keep .gitattributes itself plain

`git add .gitattributes && git commit`
Commit the rules before the secrets

`git check-attr filter -- <file>`
Check whether a file matches a rule

STATUS

`git-crypt status`
List files and their encryption state

`git-crypt status -e`
Only encrypted files

`git-crypt status -u`
Only unencrypted files

`git-crypt status -f`
Encrypt files committed in plain text

`git diff`
Plain-text diff while unlocked (diff=git-crypt)

SHARING ACCESS

`git-crypt add-gpg-user <user-id>`
Grant a GPG key access and commit

`git-crypt add-gpg-user -n <user-id>`
Grant access without committing

`git-crypt add-gpg-user --trusted <user-id>`
Skip the GPG web-of-trust check

`git-crypt export-key <file>`
Write the symmetric key to a file

`gpg --import <pubkey>`
Import a colleague's public key first

LOCK AND UNLOCK

`git-crypt unlock`
Decrypt using your GPG key

`git-crypt unlock <keyfile>`
Decrypt using an exported key

`git-crypt lock`
Re-encrypt files in the working tree

`git clone <url> && cd <repo> && git-crypt unlock`
Clone, then unlock