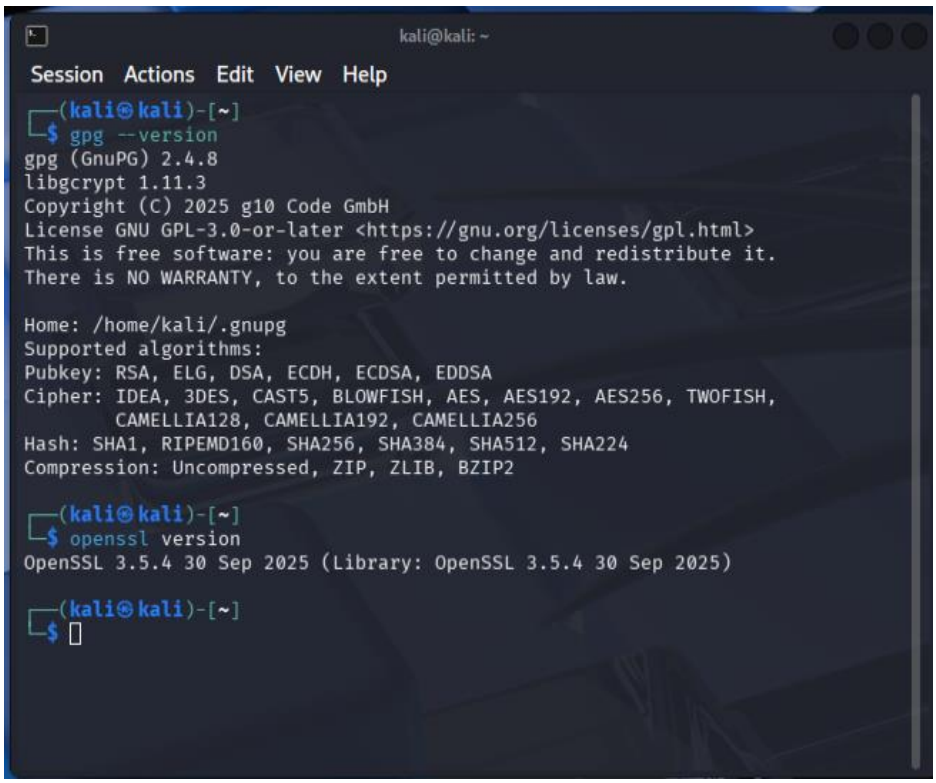
1. Sprawdzenie środowiska
   gpg --version
   openssl version



2. Generowanie kluczy

gpg --full-generate-key

3. Sprawdzenie kluczy

gpg --list-keys

gpg --list-secret-keys

```
┌──(kali㉿kali)-[~]
└─$ gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
/home/kali/.gnupg/pubring.kbx
------------------------------
pub   rsa3072 2026-01-21 [SC]
      8ECA72463FCA56E2C84C99826D57053D80290030
uid           [ultimate] Marcel (comment) <goska@gmail.com>
sub   rsa3072 2026-01-21 [E]


┌──(kali㉿kali)-[~]
└─$
```

```
┌──(kali㉿kali)-[~]
└─$ gpg --list-secret-keys
/home/kali/.gnupg/pubring.kbx
------------------------------
sec   rsa3072 2026-01-21 [SC]
      8ECA72463FCA56E2C84C99826D57053D80290030
uid           [ultimate] Marcel (comment) <goska@gmail.com>
ssb   rsa3072 2026-01-21 [E]


┌──(kali㉿kali)-[~]
└─$
```

4. Utworzenie pliku testowego

echo "To jest tajny plik projektu kryptograficznego" > tajne.txt

```
┌──(kali㉿kali)-[~/Documents/Projekt]
└─$ echo "To jest tajny plik projektu kryptograficznego" > tajne.txt
```

5.Szyfrowanie pliku (kluczem publicznym)

gpg --encrypt --recipient twoj@email.pl tajne.txt

```
┌──(kali㉿kali)-[~/Documents/Projekt]
└─$ gpg --encrypt --recipient goska@gmail.com tajne.txt

┌──(kali㉿kali)-[~/Documents/Projekt]
└─$ cat tajne.txt.gpg
•••;Ÿ5•?A
••!yĦ••••8{••••WHv•S!7••=jrcK••••••,•c~9yf•••d••y••••)U••7I5•1•F••
3••#••••]•z•Dg••j••••z••:•>                                    ••
                              4`g••I|p+••+••Ĥ••o□••w•I9•h\ys5Ł••r•gÆ••
{••{{_••A~XN•6 •2S••X•Ee3•\•••Wx•••••••••#j•4m••

┌──(kali㉿kali)-[~/Documents/Projekt]
└─$
```

6. Próba odczytu bez odszyfrowania

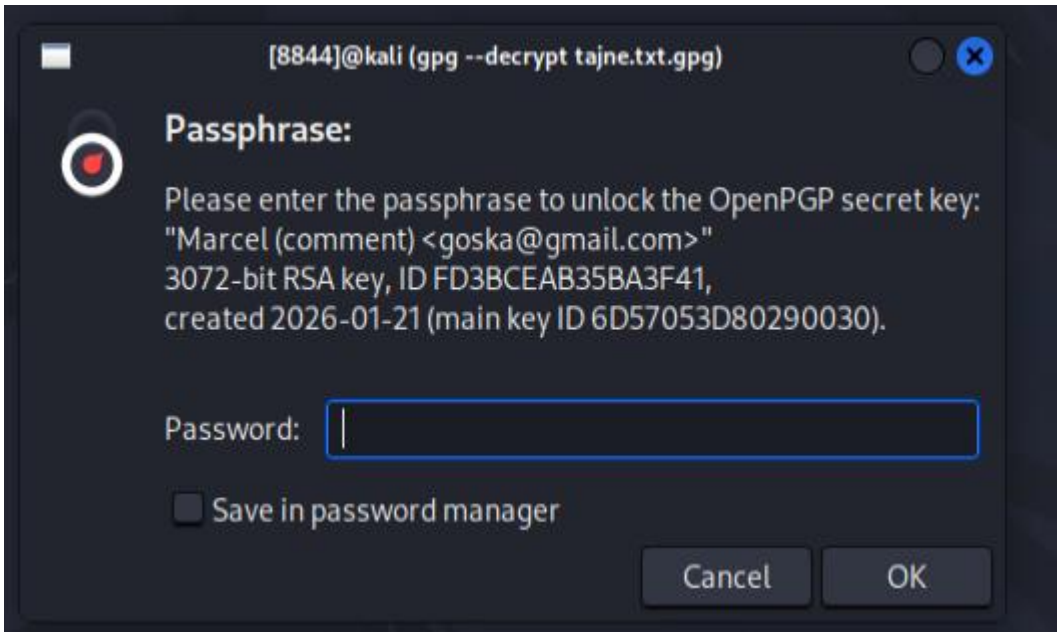
cat tajne.txt.gpg

```
┌──(kali㉿kali)-[~/Documents/Projekt]
└─$ cat tajne.txt.gpg
•••;Ÿ5•?A
••!yĦ••••8{••••WHv•S!7••=jrcK••••••,•c~9yf•••d••y••••)U••7I5•1•F••
3••#••••]•z•Dg••j••••z••:•>                                    ••
                              4`g••I|p+••+••Ĥ••o□••w•I9•h\ys5Ł••r•gÆ••
{••{{_••A~XN•6 •2S••X•Ee3•\•••Wx•••••••••#j•4m••
```

7. Deszyfrowanie pliku

gpg --decrypt tajne.txt.gpg

```
┌──(kali㉿kali)-[~/Documents/Projekt]
└─$ gpg --decrypt tajne.txt.gpg
gpg: encrypted with rsa3072 key, ID FD3BCEAB35BA3F41, created 2026-01-21
      "Marcel (comment) <goska@gmail.com>"
```




```
┌──(kali㉿kali)-[~/Documents/Projekt]
└─$ gpg --decrypt tajne.txt.gpg
gpg: encrypted with rsa3072 key, ID FD3BCEAB35BA3F41, created 2026-01-21
      "Marcel (comment) <goska@gmail.com>"
To jest tajny plik projektu kryptograficznego

┌──(kali㉿kali)-[~/Documents/Projekt]
└─$
```

8. Podpisanie Pliku

gpg --sign tajne.txt

Jawny podpis

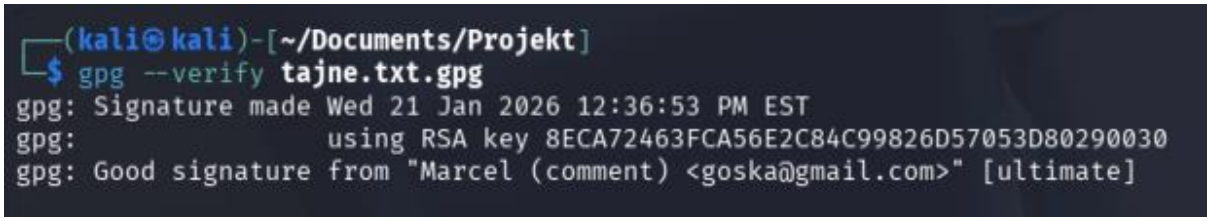
gpg --clearsign tajne.txt

```
┌──(kali㉿kali)-[~/Documents/Projekt]
└─$ gpg --sign tajne.txt
File 'tajne.txt.gpg' exists. Overwrite? (y/N) y

┌──(kali㉿kali)-[~/Documents/Projekt]
└─$ gpg --clearsign tajne.txt
```

9. Weryfikacja podpisu

gpg --verify tajne.txt.gpg



10. Test integralności

echo "Zmiana" >> tajne.txt

gpg --verify tajne.txt.gpg

Podpis cyfrowy gwarantuje integralność danych – nawet minimalna zmiana pliku powoduje niepowodzenie weryfikacji podpisu.