YubiKey and TPM2 on Linux

Hardware Security: from the command line

Valerio Vaccaro

Satoshi Spritz Connect

February 23, 2026

Valerio Vaccaro 👀

- 💻 Bitcoin Developer and Hardware Expert
- 🔥 Contributor to Bitcoin open source projects
- ⚠️ DIY hardware enthusiast
- Bitcoin and Liquid Engineer at Blockstream

Social

- 🧑‍💼 **LinkedIn** linkedin.com/in/valeriovaccaro
- 🐙 **Github** github.com/valerio-vaccaro
- **Telegram** t.me/valeriovaccaro

License

May this presentation inspire you to become more self-sovereign!

Meme


QUANDO TI DICONO CHE
SONO AL SICURO
PERCHE' HANNO UNA
CERTIFICAZIONE

Summary

- What is YubiKey and what it can do
- What is TPM2 and what it can do
- Practical command-line examples
- Comparison and use cases

Introduction

YubiKey

- Portable **Hardware Security Module (HSM)**, USB key format
- Produced by Yubico
- Cryptographic keys never leave the device
- Supports: **PIV**, **OpenPGP**, **FIDO2/WebAuthn**, **OATH** (TOTP/HOTP)

TPM2

- **Trusted Platform Module 2.0** — chip integrated on the motherboard
- TCG (Trusted Computing Group) standard
- Keys and data "sealed" to hardware/firmware state
- Used for: **secure boot**, **disk encryption**, **attestation**

Introduction

Why hardware?

- Keys not exposed in memory or on disk
- ✓ Resistance to malware and keyloggers
- YubiKey: portable across PCs | TPM2: tied to the machine

YubiKey — Installation and info

Install ykman (YubiKey Manager CLI)

```
# Debian/Ubuntu
sudo apt install yubikey-manager

# Fedora
sudo dnf install yubikey-manager

# Via pip
pip install --user yubikey-manager
```

YubiKey — Installation and info

Device information

```
# List connected YubiKeys
ykman list --serials

# Detailed info
ykman info

# Info for specific device
ykman --device 01234567 info
```

YubiKey — PIV (Smart Card)

What is PIV

- **Personal Identity Verification** — NIST standard for smart cards
- Certificate slots: **9a** (auth), **9c** (signing), **9d** (key mgmt), **9e** (card auth)
- Used for: SSH login, signing, encryption

Generate PIV key and certificate

```
# Generate ECC P-256 key in slot 9a
ykman piv keys generate --algorithm ECCP256 9a pubkey.pem

# Generate self-signed certificate
ykman piv certificates generate --subject "CN=SSH Key" 9a pubkey.pem

# Export public key for SSH
ykman piv keys export 9a pubkey.pem
ssh-keygen -D /usr/lib/opensc-pkcs11.so -e
```

YubiKey — PIV: PIN and key management

Change PIN and Management Key

```
# Change PIN (default: 123456)
ykman piv access change-pin --pin 123456 --new-pin NEW_PIN

# Change Management Key (generate random, protect with PIN)
ykman piv access change-management-key --generate --protect

# Full PIV reset (warning: erases everything!)
ykman piv reset
```

Verify certificates

```
# List certificates in slots
ykman piv certificates list
```

YubiKey — OpenPGP / GPG

Configure GPG on YubiKey

```
# Start card configuration
gpg --card-edit

# Useful commands in card-edit:
# admin      -> administrative options
# name       -> name on token
# url        -> URL for key recovery
# key-attr   -> key algorithm (e.g. rsa4096 or ed25519)
# generate   -> generate keys on YubiKey
```

YubiKey — OpenPGP / GPG

Import existing key onto YubiKey

```
# 1. Generate key on PC (or you already have keyid)
gpg --full-generate-key

# 2. Transfer subkeys to YubiKey
gpg --edit-key KEYID
# gpg> key 1
# gpg> keytocard
# gpg> key 2
# gpg> keytocard
# ...
```

YubiKey — GPG: touch and PIN

Touch requirement and PIN management

```
# Require touch for authentication key
ykman openpgp keys set-touch aut on

# Require touch for signing
ykman openpgp keys set-touch sig on

# Change user PIN
ykman openpgp access change-pin --pin OLD --new-pin NEW

# Change Admin PIN (min 8 characters)
ykman openpgp access change-admin-pin --admin-pin OLD --new-admin-pin NEW
```

YubiKey — GPG: touch and PIN

Using GPG with YubiKey

```
# Sign
gpg --sign document.pdf

# Decrypt (with PIN/touch)
gpg --decrypt document.pdf.gpg
```

YubiKey — SSH with PIV/GPG

SSH with PIV (PKCS#11)

```
# Add PIV key to SSH agent
ssh-add -s /usr/lib/opensc-pkcs11.so

# Or in ~/.ssh/config
Host *
    PKCS11Provider /usr/lib/opensc-pkcs11.so
```

YubiKey — SSH with PIV/GPG

SSH with GPG (gpg-agent)

```
# Enable SSH support in gpg-agent (~/.gnupg/gpg-agent.conf)
enable-ssh-support

# Restart agent
gpgconf --kill gpg-agent

# List keygrip for SSH
gpg --list-keys --with-keygrip
```

YubiKey — OATH (TOTP/HOTP)

Configure OATH on YubiKey

```
# Add TOTP credential from URI
ykman oath add -t "Account:user@example.com" \
  "otpauth://totp/Account:user?secret=JBSWY3DPEHPK3PXP"

# List credentials
ykman oath list

# Show TOTP code (requires touch)
ykman oath code "Account:user@example.com"

# Remove credential
ykman oath delete "Account:user@example.com"
```

YubiKey — OATH (TOTP/HOTP)

FIDO2 / WebAuthn

- Passwordless login on compatible sites
- Phishing resistant
- Configurable from browser or `ykman fido`

TPM2 — Basic concepts

What is the TPM

- Dedicated chip on the motherboard
- 🔒 Keys generated and used only inside the TPM
- **PCR** (Platform Configuration Registers): hash of firmware, bootloader, kernel
- **Sealing**: data encrypted bound to specific PCR values

Why it's useful

- 💾 Automatic LUKS unlock if boot is intact
- 🔑 Secure Boot + measured boot
- 🛡️ Attestation: proves system state

TPM2 — Installation and verification

Install tpm2-tools

```
# Debian/Ubuntu
sudo apt install tpm2-tools tpm2-abrmd

# Fedora
sudo dnf install tpm2-tools

# Verify TPM presence
tpm2_getcap properties-fixed
tpm2_getcap properties-variable
```

TPM2 — Installation and verification

Verify TPM is working

```
# Basic test: generate random number
tpm2_getrandom 32 | xxd

# TPM info
tpm2_getcap manufacturers
```

TPM2 — Create and use keys

Hierarchy and primary keys

```
# Create context for primary key (ECC)
tpm2_createprimary -C e -g sha256 -G ecc -c primary.ctx

# Create child key (key to use)
tpm2_create -C primary.ctx -g sha256 -G ecc -u key.pub -r key.priv

# Load key into TPM
tpm2_load -C primary.ctx -u key.pub -r key.priv -c key.ctx

# Persist in NVRAM slot (optional)
tpm2_evictcontrol -C o -c key.ctx 0x81010002
```

TPM2 — Create and use keys

Sign with TPM key

```
# Sign hash of a file
tpm2_sign -c key.ctx -g sha256 -o sig.bin file.txt
```

TPM2 — PCR and sealing

What are PCRs

- 24 registers (0–23) with hash of boot components
- PCR 0–7: firmware, bootloader
- PCR 8: UEFI events
- PCR 9: bootloader
- If boot changes -> PCRs change -> sealed data cannot be decrypted

Seal/Unseal with PCR

```
# Create policy based on PCR 0,1,2
tpm2_createpolicy --policy-pcr -l sha256:0,1,2 -f policy.pcr

# Seal data (encrypt bound to PCR)
echo "secret" | tpm2_create -C primary.ctx -G keyedhash -u key.pub -r key.pri
  -L policy.pcr -i - -o sealed.dat
```

TPM2 — Clevis and LUKS

What is Clevis

- Framework for automatic decryption
- **Pins**: tpm2, tang, sss, ...
- Binds secrets to policy (e.g. PCR, Tang server)

Clevis + TPM2: encrypt data

```
# Encrypt file bound to TPM (default PCR)
echo "sensitive data" | clevis encrypt tpm2 '{}' > encrypted.jwe

# Decrypt (works only on same PC, same boot state)
clevis decrypt < encrypted.jwe
```

TPM2 — Clevis and LUKS

Clevis + TPM2: custom PCR policy

```
# Bind to PCR 0 and 7 (e.g. Secure Boot)
clevis encrypt tpm2 '{"pcr_ids":"0,7"}' < plaintext > encrypted.jwe

# With specific algorithm
clevis encrypt tpm2 '{"hash":"sha256","key":"ecc"}' < plaintext > encrypted.jw
```

TPM2 — LUKS with Clevis (automatic unlock)

Add Clevis slot to LUKS

```
# Partition already encrypted with LUKS
sudo clevis luks bind -d /dev/sda3 tpm2 '{"pcr_ids":"0,1,2,3,4,5,6,7"}'

# New LUKS volume with Clevis
sudo cryptsetup luksFormat /dev/sda3 --type luks2
sudo clevis luks bind -d /dev/sda3 tpm2 '{}'
```

Automatic unlock at boot

- With `clevis luks bind`, initramfs can unlock the partition
- 💾 No password if boot is not altered
- If firmware/kernel change -> password or recovery key required

```
# Verify slots
sudo clevis luks list -d /dev/sda3
```

TPM2 — Complete LUKS + Clevis example

Setup on Debian/Ubuntu

```
# Install packages
sudo apt install clevis clevis-luks clevis-systemd

# Bind Clevis to existing LUKS
sudo clevis luks bind -d /dev/nvme0n1p3 tpm2 '{}'

# Update initramfs
sudo update-initramfs -u -k all
```

Recovery

- Keep a LUKS passphrase or recovery key
- If TPM doesn't unlock (hardware/firmware changed):
  - Enter passphrase manually
  - Or use `clevis luks unlock` with Tang/SSS if configured

YubiKey vs TPM2 comparison

YubiKey

Pro	Con
Portable across PCs	Can be lost
PIV, GPG, FIDO2, OATH	Cost (~50–100€)
Touch for confirmation	Requires USB/NFC
Widely used for SSH/GPG	

YubiKey vs TPM2 comparison

TPM2

Pro	Con
Integrated in PC	Not portable
Automatic LUKS unlock	Tied to single machine
No additional cost	Less flexible for GPG/SSH keys
Measured boot, attestation	

Practical use cases

When to use YubiKey

- GPG keys for signing/encryption
- SSH from multiple workstations
- FIDO2/WebAuthn for web login
- TOTP/HOTP (2FA) on a single device
- Remote work with portable keys

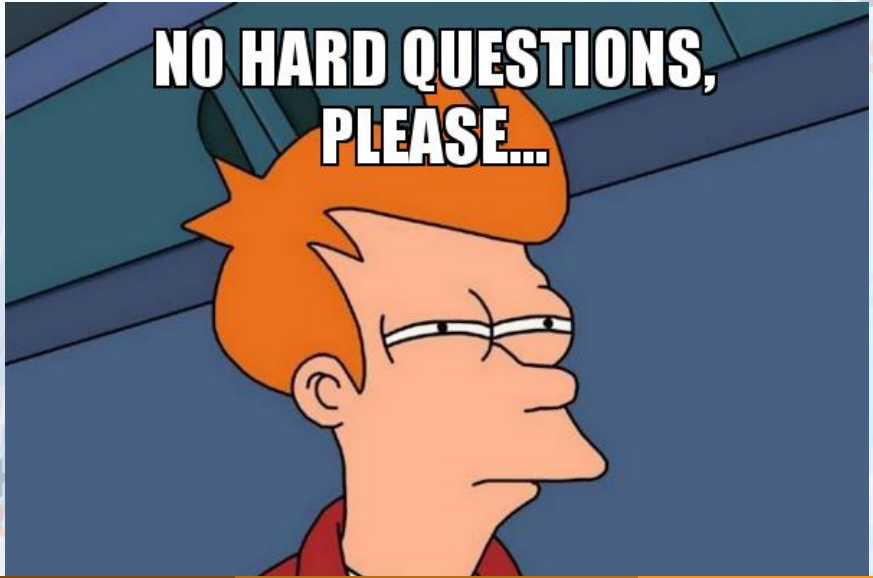
When to use TPM2

- Automatic encrypted disk unlock
- Secure Boot + measured boot
- Server/VM without human interaction at boot
- Keys bound to system integrity

Conclusions

- 🔑 **YubiKey**: Portable HSM for GPG, SSH, FIDO2, OATH — ideal for keys that travel with you
- 💻 **TPM2**: Integrated chip for sealing, automatic LUKS, attestation — ideal for protecting the single machine
- 🛡️ Both keep keys out of memory and disk
- 🔗 Can be used together: YubiKey for identity, TPM2 for disk unlock

? Questions



Bibliography

- **Yubico**: docs.yubico.com — YubiKey Manager, PIV, OpenPGP
- **TCG TPM 2.0**: Trusted Platform Module specifications
- **Clevis**: github.com/latchset/clevis — Automated decryption
- **tpm2-tools**: github.com/tpm2-software/tpm2-tools

Online resources

- developers.yubico.com
- tpm2-software.github.io
- blog.dowhile0.org - LUKS + TPM2

Satoshi Spritz Project

- Federation of local Bitcoiner groups
- Free and privacy-oriented events
- BITCOIN ONLY
- Weekly Satoshi Spritz Connect online
- Oriented toward self-sovereign learning

Links

- satoshispritz.it
- t.me/SatoshiSpritzConnect

Officine Bitcoin

- 🤝 Italian Bitcoiners community, fully free
- 🤖 BITCOIN ONLY
- 🎓 Focus on education and project development
- 📋 Projects: Bitcoin nodes, hardware wallet, open source, Debian, mnemonics, ...

Links

- officinebitcoin.it