
SBA
Research

The TPM and You!

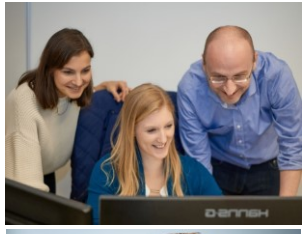
How to actually make use of your TPM

Bundesministerium Innovation, Mobilität und Infrastruktur
Bundesministerium Wirtschaft, Energie und Tourismus
FFG Forschung wirkt.
wirtschaftsagentur wien
Für die Stadt Wien
European Commission
FWF Österreichischer Wissenschaftsfonds
netidee FÖRDERUNGEN

whoami

- IT Security Consultant at SBA Research
  - Penetration testing, SDLC, DevSecOps, Linux, CI Security, Cryptography, …
- Previously
  - Mathematician, Dev, SysAdmin, Security Officer, University Teacher

mtausig@sba-research.org



SBA Research

Forschung & Beratung unter einem Dach

SBA Research
Professional Services

Security Governance

- Security Governance Lagebild
- ISMS / ISO27001
- Compliance (NIS-2, CRA, DORA, etc.)
- Risikomanagement
- Audit

Cyber Defense

- Cyber Security Lagebild
- Penetrationstests der (Cloud) Infrastruktur
- SWIFT CSP Audit
- Social Engineering / Spear Phishing
- Red – Blue – Purple Teaming

Security Schulungen

- Cyber Security Essentials
- Secure Coding Fundamentals
- Web Application Hacking
- Cyber Defense / Hacking
- Security Awareness

Software Security

- Sicherer Softwareentwicklungsprozess
- Threat Modeling
- Application Pentesting
- Mobile Application Pentesting
- CI/CD Audit

Treten Sie unserer MeetUp Gruppe bei!

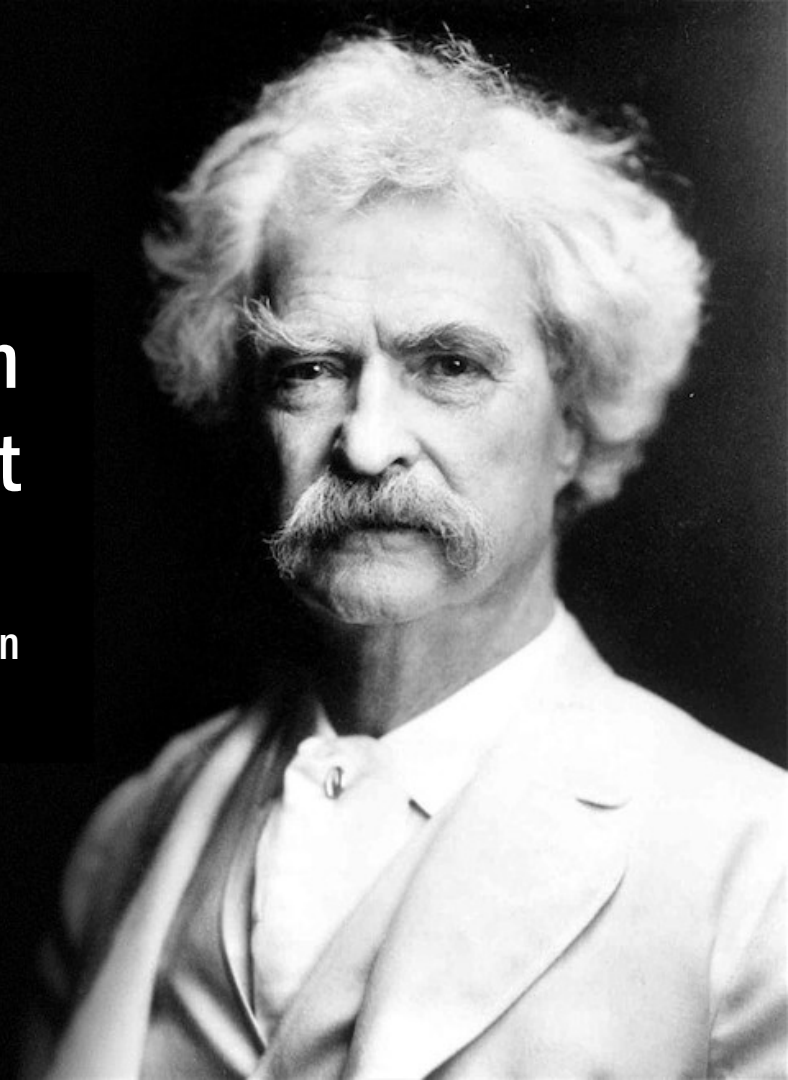
https://www.meetup.com/Security-Meetup-by-SBA-Research/

SBA Research

Motivation

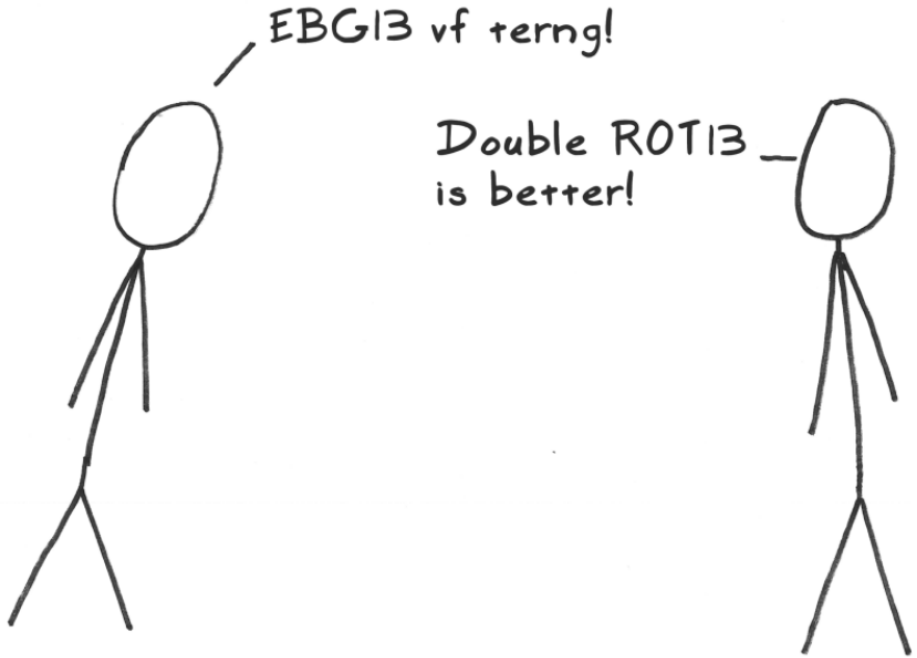
Every problem in cryptography can be reduced to a key management problem.

-- Abraham Lincoln



Hard Problem in Cryptography

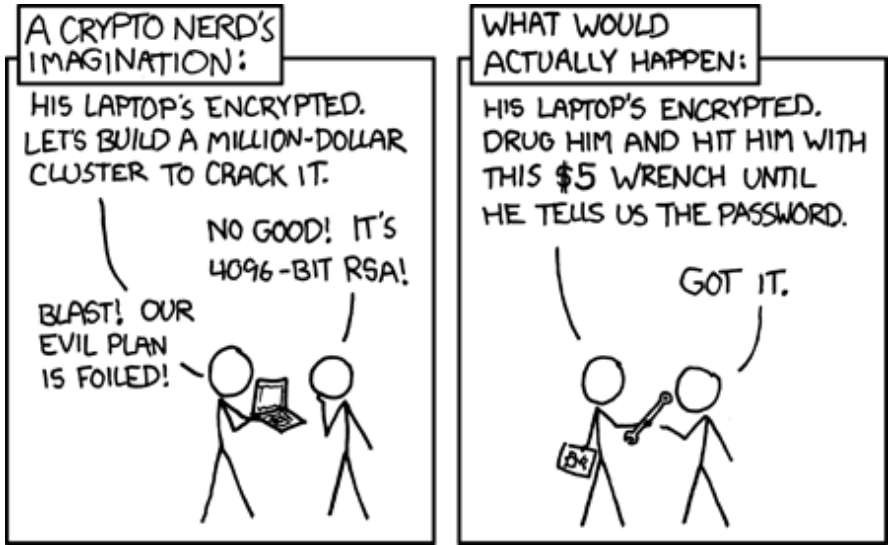
- Common discussions about cryptography:
  - Algorithms to use
  - Key lengths
  - Cipher suites
  - Implementations



https://www.moserware.com/assets/stick-figure-guide-to-advanced/A%20Stick%20Figure%20Guide%20to%20the%20Advanced%20Encryption%20Standard%20%28AES%29.pdf

Hard Problem in Cryptography

- Common problem in cryptography rarely discussed: **Key management**
  - Every algorithm uses secrets
  - Where to store them?
  - How to restrict access?



https://xkcd.com/538/

Storing Keys

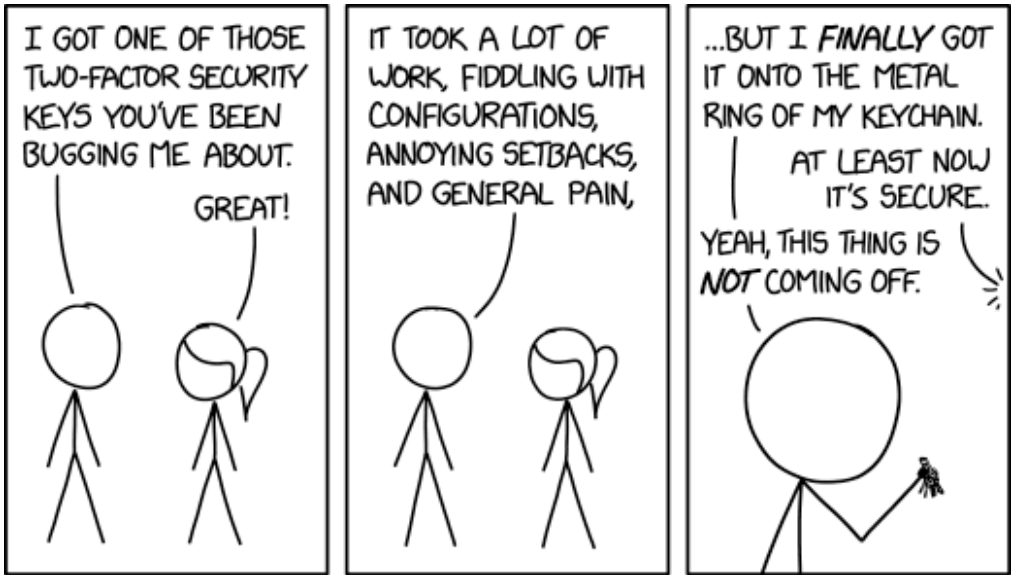
- Where to store the secrets, your application uses (with varying degrees of insecurity)?
  - Source code
  - Plaintext config file
  - Password encrypted config file
  - Fancy secret management tool (K8S Secrets, Vault, …)
- Common problem: Cleartext extraction always possible
  - (Sometimes easier, sometimes harder)
  - For cryptographic keys cleartext form is not required

Storing Keys

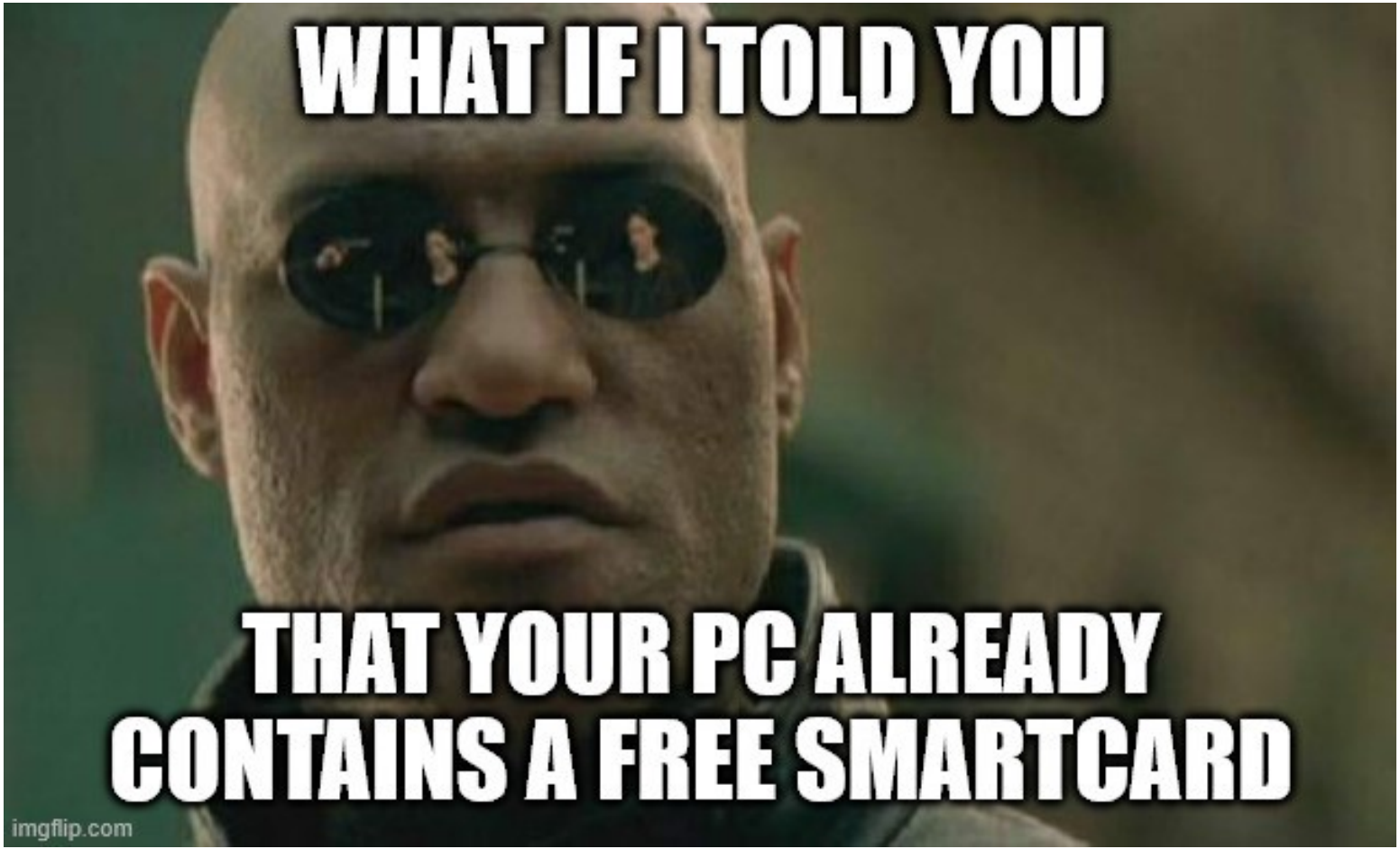
- If we want to prevent our keys from being extractable, we need some dedicated hardware
- Keys can be used, not read
- Varying degree of physical protection
- Brute-Force protection
- Smartcard, token, HSM, …

Cryptographic Hardware

- Great for security
  - not necessarily for usability
- Expensive?
- Backups?
- Implementations?
- Speed?
- Storage capacity?

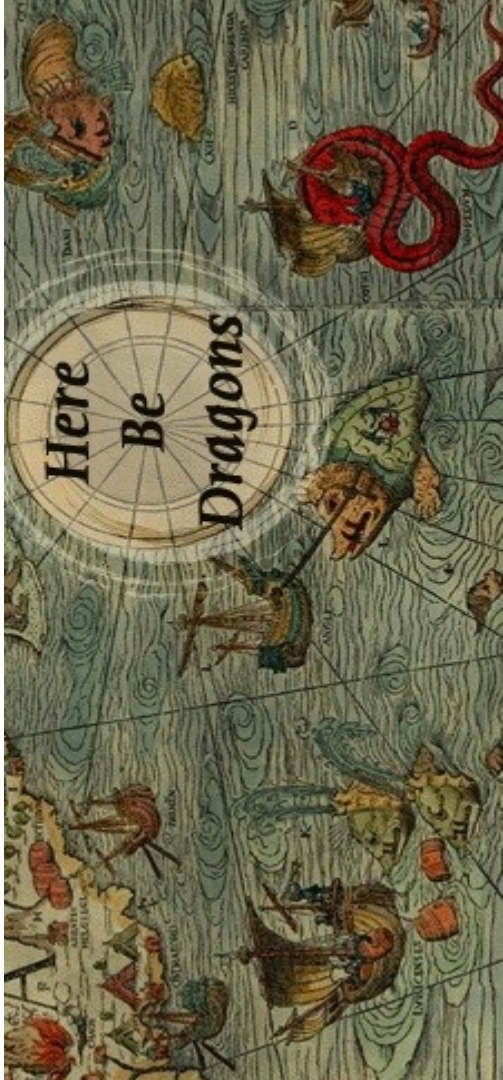


https://xkcd.com/2522/


WHAT IF I TOLD YOU
THAT YOUR PC ALREADY
CONTAINS A FREE SMARTCARD
imgflip.com

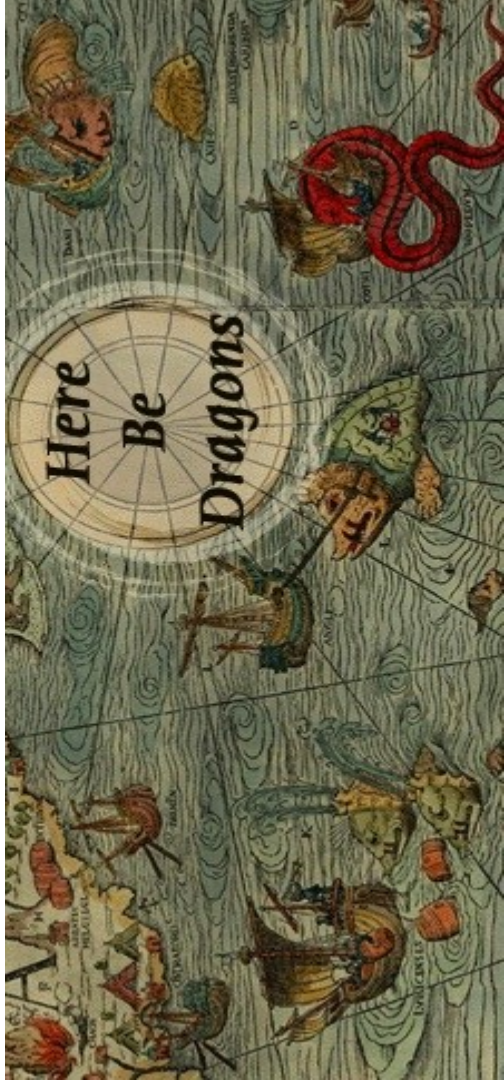
Brief History

- TCPA (Trusted Computing Platform Alliance) founded in 1999 by Microsoft, Intel, IBM et.al.
- Unofficial mission statement: “Make China pay for Windows”
- After much backlash renamed (TCG) and scaled down
- First TPM chips deployed in 2003
- v1.2 spec published in 2009, v2 in 2014



Brief History

- Windows 7 was the first common OS to make use of a TPM for security (Bitlocker)
- Steady increase in consumer mainboards containing it
- Microsoft mandates TPMv2 for Windows 11 (2021)
- Very few devices in use today which do not contain a TPM chip



What?

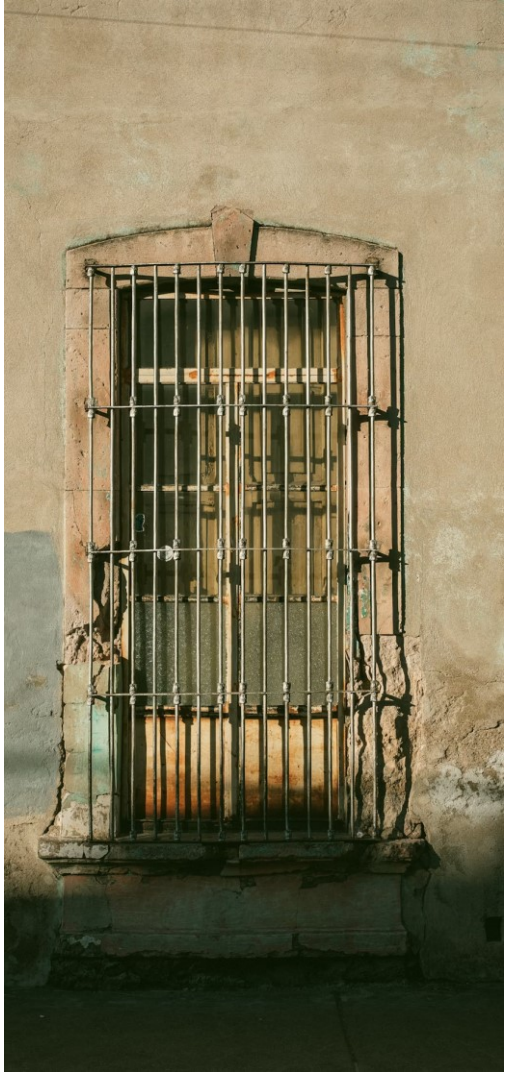
- A TPM
  - is a discrete chip on your mainboard or an equivalent interface provided by the CPU (fTPM, iTPM)
  - can generate and store keys
  - can perform cryptographic computations
  - never exposes stored keys in plaintext
  - can often be upgraded via the firmware
  - is cheap

What Not?

- A TPM
  - is (usually) not secured against physical attacks
  - is slow
  - has very limited storage space (~64kB)

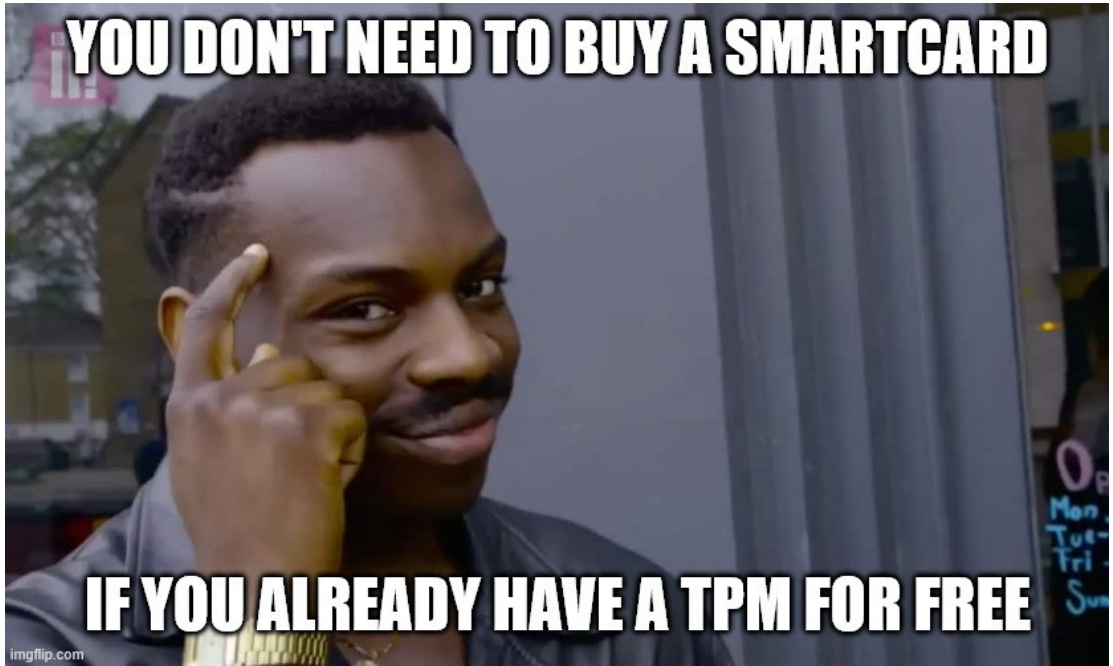
Scope

- TPMs have a lot of diverse usages, we cannot cover all of them. Out of scope:
  - Political controversies about TPMs and *trusted computing* (some deserved, some not)
  - Secure Boot / PCR
  - Attestation
  - More general: sysadmin stuff
  - Similar implementations on other platforms (e.g., ARM TrustZone, Apple Keychain, …)
  - TPM < v2



Scope

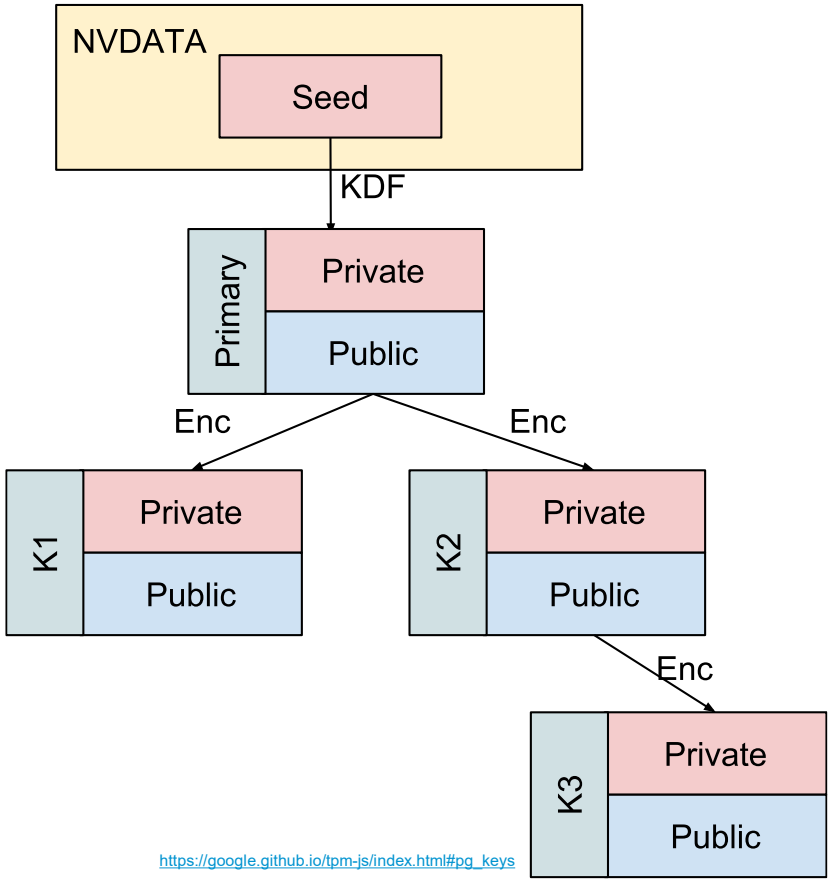
- In scope rather: **How can we use the TPM to secure our applications?**



Concepts

Hierachies

- All cryptographic keys are organized in *hierarchies*, forming a tree
- Each key is encrypted by its parent
- Each *Primary Key* is derived from a fixed, persistent *seed* and a *template*



https://google.github.io/tpm-js/index.html#pg_keys

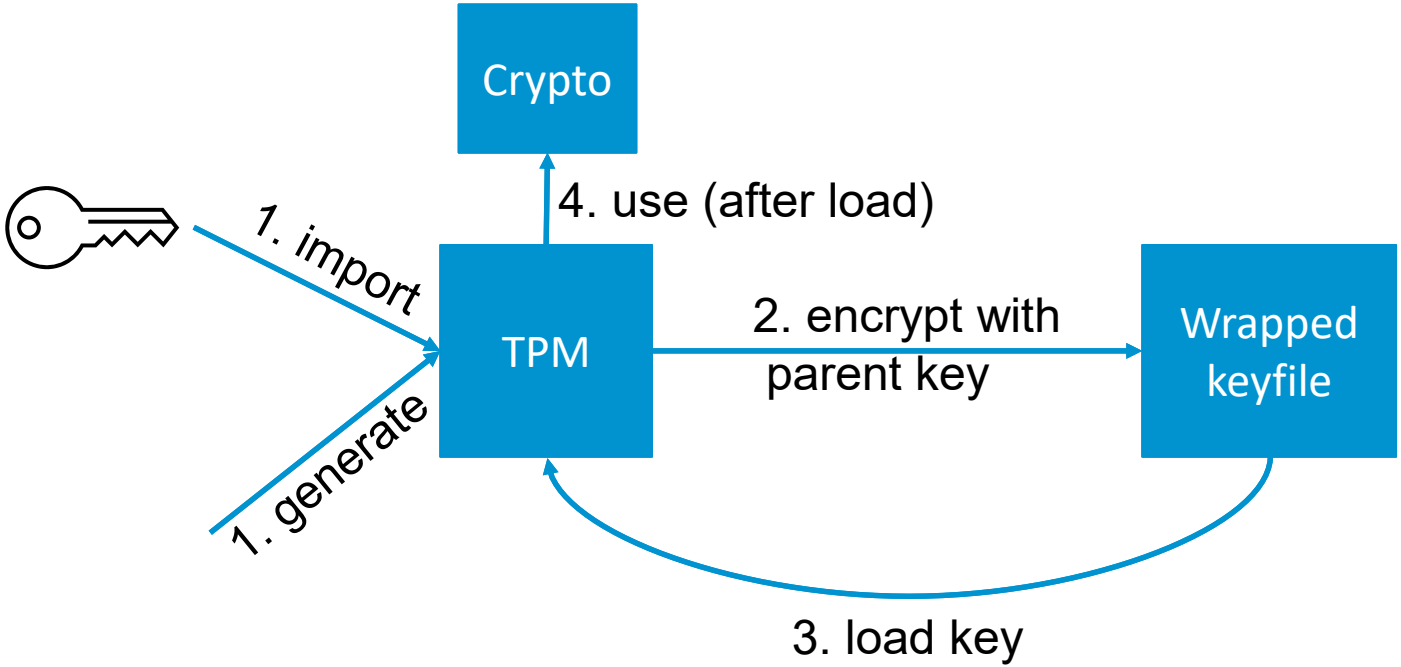
Hierarchies

There are 4 hierarchies available:

Hierarchy	Properties
Endorsement hierarchy	Seed never changes; primary key certified by manufacturer
Platform hierarchy	Seed reset and controlled by *platform owner* (OEM)
Owner hierarchy/storage hierarchy	**Seed can be reset and controlled by device owner (user or IT department)**
Null hierarchy	Ephemeral hierarchy; seed reset on reboot

Storage

Keys are (usually) not stored in the TPM



Authorization

- Each key (primary or child) can have an (optional) *Authorization Policy* defining conditions for accessing a key
  - Authentication with a secret (password, signature)
  - Time-based rules
  - Usage-based rules
  - PCR state
  - …
- TPMs have a brute-force protection
  - Whole device blocks for a defined timespan after consecutive unsuccessful authentication attempts

OTP Application

TOTP

One-time password authenticator (OTP) 0

Use a one-time password authenticator on your mobile device or computer to enable two-factor authentication (2FA).

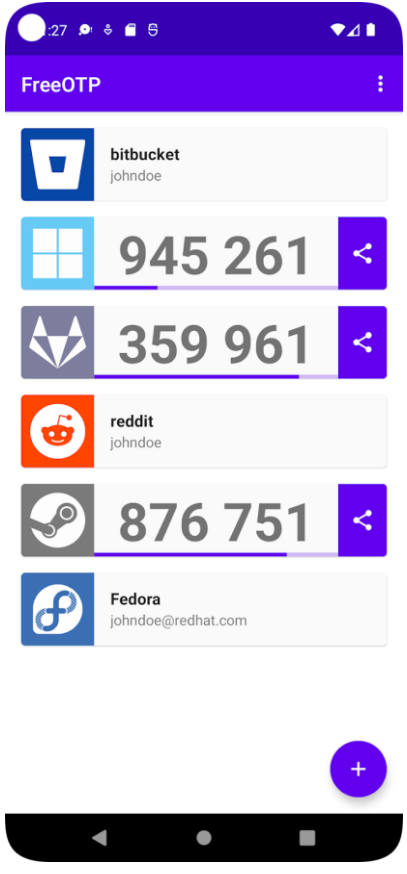
Can't scan the code?

To add the entry manually, provide the following details to the application on your phone.

Account: code.sba-research.org:mtausig@sba-research.org
Key: `2GL2 BIAZ UNY2 5Q5E RTGK Z42X AWYA LNIY`
Time based: Yes

Enter verification code

Register with two-factor app | Cancel



Under The Hood

`otpauth://totp/mtausig?secret=2GL2BIAZUNY25Q5ERTGKZ42XAWYALNIY&issuer=SBA&otp-algorithm=SHA1&otp-digits=6`

Metadata

- Label
- Issuer
- Hash alg. (SHA-1)
- Timeframe (30s)
- Digits (6)

Secrets

- OTP-Secret (BASE32)

Moving Factor

- Timestamp (TOTP)
- Counter (HOTP)

HMAC

1637409809a679dc698207310c8c7fc07290d9e5

520489

https://www.ietf.org/archive/id/draft-linuxgemini-otpauth-uri-02.html

TOTP Security

- Key needs to be s
- What if it gets lea
- Mobile applicatio
  storage (Keystore,
- Desktop PCs rarel
  - Insecure storag
  - Semisecure sto
  - Need for extra

Backup

Whenever you are storing unexportable keys on a hardware device, you need a plan B!

- Is THIS key essential or can I just recreate a new one?
  - Import software key vs. generate on device
- Secure offline backup or second active key

Live Demonstration

https://gitlab.com/mtausig/totpm/

Photo by Victor on Unsplash

OTP Calculation

```go
type Hasher interface {
    ComputeHash(data []byte) ([]byte, error)
}
```

```go
func HOTP(hasher Hasher, counter uint64, config HOTPconfig) (string, error) {
    var counterBytes []byte
    counterBytes = binary.BigEndian.AppendUint64(counterBytes, counter)
    hash, err := hasher.ComputeHash(counterBytes)
    if err != nil {
        return "", err
    }
    sNum, err := dynamicTruncation(hash)
    if err != nil {
        return "", err
    }
    return computeHOTP(sNum, config.Digits), nil
}
```

Software Key

```
type SimpleHasher struct {
    Key []byte
}

func (h SimpleHasher) ComputeHash(data []byte) ([]byte, error) {
    mac := hmac.New(sha1.New, h.Key)
    mac.Write(data)
    ret := mac.Sum(nil)
    return ret, nil
}

func FromBytes(key []byte) SimpleHasher {
    return SimpleHasher{
        Key: key,
    }
}
```

Configuration

```
type AccountDb []Account

const ConfigFileName = "totpm.conf"

func (db AccountDb) Store() error {
    jsonBin, err := json.Marshal(db)
    […]
    configFile := filepath.Join(configDir, ConfigFileName)
    err = os.WriteFile(configFile, jsonBin, 0600)
    […]
}

func ReadConfig() (AccountDb, error) {
    […]
    config, err := os.ReadFile(configFile)
    […]
    var ret AccountDb
    err = json.Unmarshal(config, &ret)
    […]
}
```

Software Key

```
$ ./totpm -add –tpm=false
otpauth://totp/simple?secret=GEZDGNBVGY3TQOJQGEZDGNBVGY3TQOJQ
$ ./totpm -otp simple
OTP of 'simple': 510985
$ cat ~/.config/totpm.conf | jq .
[
  {
    "Config": {
      "Label": "simple",
      "Digits": 6,
      "Period": 30,
      "Issuer": ""
    },
    "Hasher": {
      "Key": "MTIzNDU2Nzg5MDEyMzQ1Njc4OTA="
    }
  }
]
$ echo -n "GEZDGNBVGY3TQOJQGEZDGNBVGY3TQOJQ" | base32 -d
12345678901234567890
$ echo "MTIzNDU2Nzg5MDEyMzQ1Njc4OTA=" | base64 -d
12345678901234567890
```

TPM Key

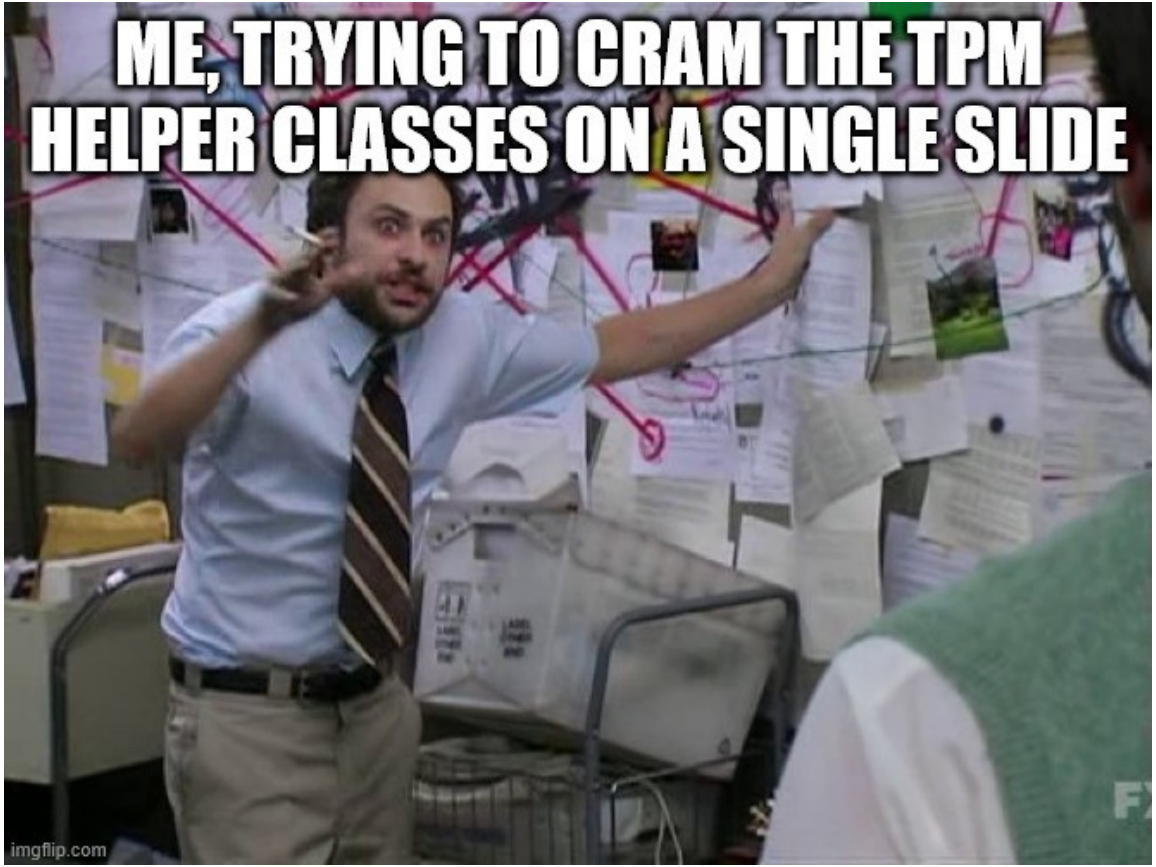
```
type TpmHasher struct {
    KeyPublicPart  []byte
    KeyPrivatePart []byte
}

func FromBytes(key []byte) (TpmHasher, error) {
    tpmTransport, err := tpmConnection()
    […]
    defer tpmTransport.Close()
    primary, err := primaryKey(tpmTransport)
    […]
    defer flushObject(tpmTransport, primary.ObjectHandle)
    hmacPublic, hmacPrivate, err := importHMACKey(tpmTransport, key, *primary, SHA1)
    […]
    hmacPublicSerialized := tpm2.Marshal(hmacPublic)
    hmacPrivateSerialized := tpm2.Marshal(hmacPrivate)
    return TpmHasher{
        KeyPublicPart:  hmacPublicSerialized,
        KeyPrivatePart: hmacPrivateSerialized,
    }, nil
}
```

TPM Key

```
func (h TpmHasher) ComputeHash(data []byte) ([]byte, error) {
    […]
    defer tpmTransport.Close()
    primary, err := primaryKey(tpmTransport)
    […]
    defer flushObject(tpmTransport, primary.ObjectHandle)
    hmacPublicDeserialized, err := tpm2.Unmarshal[tpm2.TPM2B[tpm2.TPMTPublic,
*tpm2.TPMTPublic]](h.KeyPublicPart)
    […]
    hmacPrivateDeserialized, err := tpm2.Unmarshal[tpm2.TPM2BPrivate](h.KeyPrivatePart)
    […]
    hmacKey, err := loadKey(tpmTransport, *primary, *hmacPublicDeserialized,
*hmacPrivateDeserialized)
    […]
    defer flushObject(tpmTransport, hmacKey.ObjectHandle)
    tpmMac, err := calculateHMAC(tpmTransport, hmacKey, data)
    […]
    return tpmMac, nil
}
```

TPM Key: Lowlevel



TPM Key: Lowlevel

```
func importHMACKey(transport transport.TPM, key []byte, primary tpm2.CreatePrimaryResponse,
algorithm HashType) (tpm2.TPM2BPublic, tpm2.TPM2BPrivate, error) {
    […]
    hmacTemplate := tpm2.TPMTPublic{
        Type:    tpm2.TPMAlgKeyedHash,
        NameAlg: tpm2.TPMAlgSHA256,
        ObjectAttributes: tpm2.TPMAObject{
            FixedTPM:            false,
            FixedParent:         false,
            SensitiveDataOrigin: false,
            UserWithAuth:        true,
            SignEncrypt:         true,
        },
        AuthPolicy: tpm2.TPM2BDigest{},
        Parameters: tpm2.NewTPMUPublicParms(tpm2.TPMAlgKeyedHash,
            &tpm2.TPMSKeyedHashParms{
                Scheme: tpm2.TPMTKeyedHashScheme{
                    Scheme: tpm2.TPMAlgHMAC,
                    Details: tpm2.NewTPMUSchemeKeyedHash(tpm2.TPMAlgHMAC,
                        &tpm2.TPMSSchemeHMAC{
                            HashAlg: algId,
[…]
```

TPM Key

```
$ ./totpm -add
otpauth://totp/tpm?secret=GEZDGNBVGY3TQOJQGEZDGNBVGY3TQOJQ"
$ ./totpm -otp simple; ./totpm -otp tpm
OTP of 'simple': 320145
OTP of 'tpm': 320145
$ cat ~/.config/totpm.conf | jq .
[
  {[…]},
  {
    "Config": {
      "Label": "tpm",
      "Digits": 6,
      "Period": 30,
      "Issuer": ""
    },
    "Hasher": {
      "KeyPublicPart":
"ADAACAALAAQAQAAAAAUABAAgv/QipY6i+SilMXLY5z/pW9TvECGTVQAisHFBf8DpwtQ=",
      "KeyPrivatePart": "AHIAIGh9YyVDdiNiZ[…]"
    }
  }
]
```

Different Host

```
$ # Copy totpm.conf to  ~/.config/

$ ./totpm -otp simple
OTP of 'simple': 555500

$ ./totpm -otp tpm
ERROR: Could not calculate OTP: TPM_RC_INTEGRITY (parameter 1): integrity check failed
```

Wrapup

Benefits

- Was it worth it?
  - This protects us from an attacker having one-time read access to our home directory
  - Our desktop PC becomes a true *something you have* authentication factor
- Room for improvement?
  - Add a PIN to defend against attacker with *code execution* privileges

Benefits

- Do we get perfect security?
  - No
- Is it better than a plaintext key in a config file?
  - Yes, by far!

More Ideas

- Hardware RNG
- Use for encryption keys (*key encryption key)*
- Use for signature keys
- Use for access keys e.g., SSH (PKCS#11 wrapper)
- Securely passing credentials to your application (systemd-creds)
- LUKS

TL;DR

- Every current PC has a free smartcard/Yubikey equivalent included
- Using it can solve many key management problems without impairing usability
- Do not expect a friendly developer XP
- Don’t let a (justified) political discussion keep your from using a TPM for a good purpose

Further Reading

- The Spec
- The Book: A Practical Guide to TPM 2.0
- 3-part blog series (German)
- Kernel developer Matthew Garrett
- Sample repository with common use cases for CLI and code (golang)
- TPM in JS with an online walkthrough
- Another presentation like this one

Questions?

Mathias Tausig

SBA Research

Floragasse 7, 1040 Wien

mtausig@sba-research.org

Matrix: @mtausig:sba-research.org

https://www.sba-research.org/appsec

Photo by Miguel Orós on Unsplash