FutureTPM
H2020 PROJECT:
Device Management Use Case

2nd Review Meeting, 18/02/2021, online

Roberto Sassu, Silviu Vlasceanu (HWDU)

roberto.sassu/silviu.vlasceanu@huawei.com

The FutureTPM project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779391.

Outline

- Use case overview
- Technology and functionality of the demonstrator
- Evaluation
- Conclusions

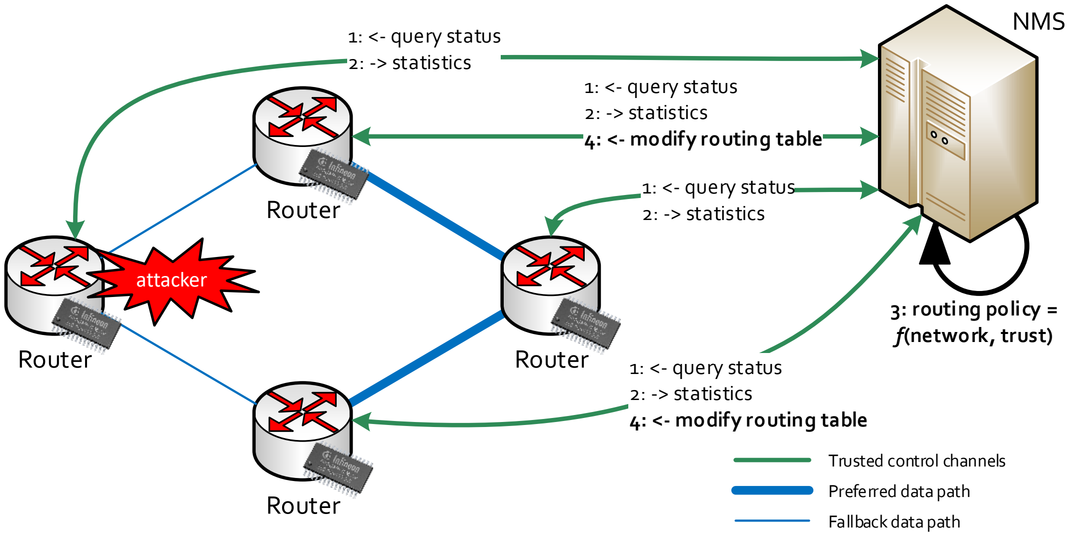
Device Management Overview

Management of enterprise network infrastructure

- Network elements (e.g. routers)
- Network Management System (NMS)
- Endpoints (e.g. laptops, servers)

Operations of the network infrastructure

- NMS queries the routers to obtain their status
- NMS sends configuration commands to the routers in response to certain events (e.g. router offline)



Why We Need FutureTPM

- Weak device identification
  - Device key is stored in the device storage unprotected
- Software integrity is not monitored
  - A compromised router could ignore management commands sent by the NMS
  - Without detection by the NMS, an attacker can continue to perform his actions
- Data integrity and confidentiality is not monitored
  - Data is often stored in plain text and integrity is not verified
  - Data can be accessed by the device even when compromised
- Telco equipment has a very long lifespan (>10 years)
  - Existing products must be able to switch to QR algorithms when quantum computing becomes practical

Main Artifacts Shown in the demo

- New network management solution fulfilling the strong security requirements defined in WP1 [D6.5]
- Advanced technology at OS level for remote attestation (CIV) [D6.3]
- Virtualization components enhanced to work with QR-TPM (QEMU, SeaBIOS, Linux kernel, …) [D6.3]
- Software TPM [D5.3]

Device Management Demonstrator Features

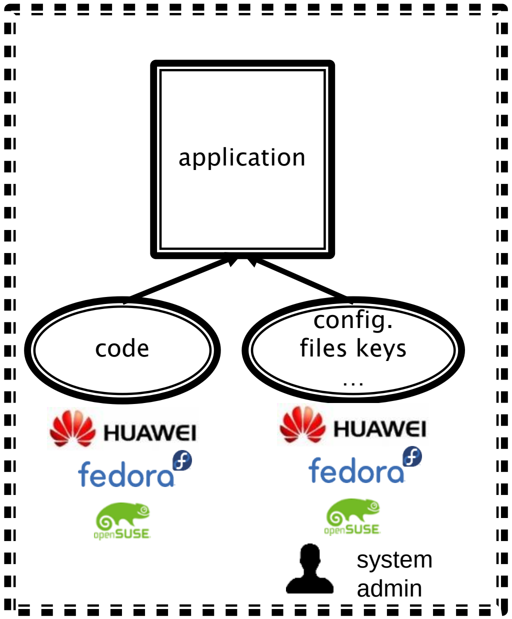
- Strong hardware-based identification
- Continuous monitoring of system and data integrity
- Secure Zero Touch Provisioning
- Integration with QR-TPM and use of QR algorithms
- Trust-aware routing decisions

Strong Device Identification

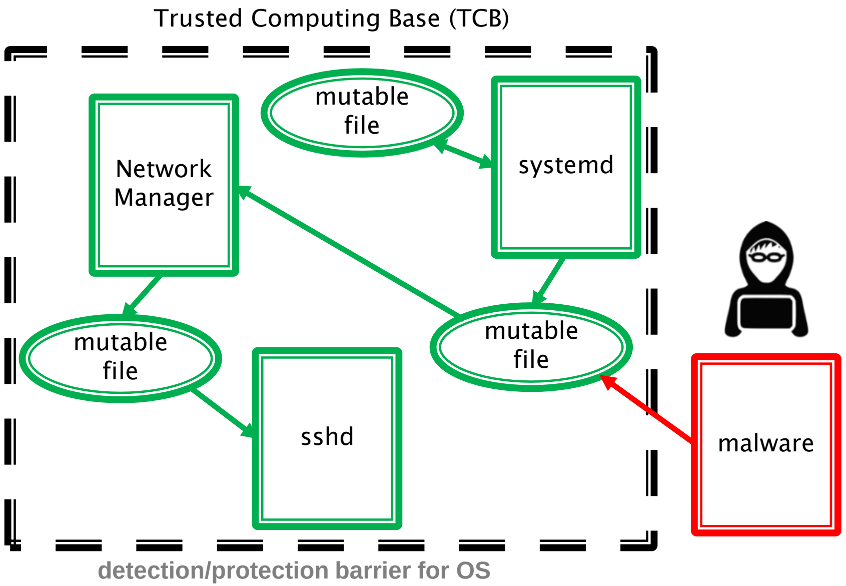
- Common issue in network management
  - The identification key is stored in the device storage unprotected
  - It is easy to move the key to another device to impersonate a legitimate one

- TPM solves this issue
  - TPM keys are never in plaintext outside the TPM and are bound to a specific TPM
  - TPM is usually soldered in the device mainboard and cannot be moved to another device
  - TPM can be uniquely identified from its Endorsement Key (EK)
  - A certificate for the EK (EK credential) is provided by TPM vendors, also via offline mechanisms (e.g. email)

Integrity Protection and Detection



load-time
integrity



run-time
integrity

offline
integrity

Comprehensive Integrity Verification (CIV)

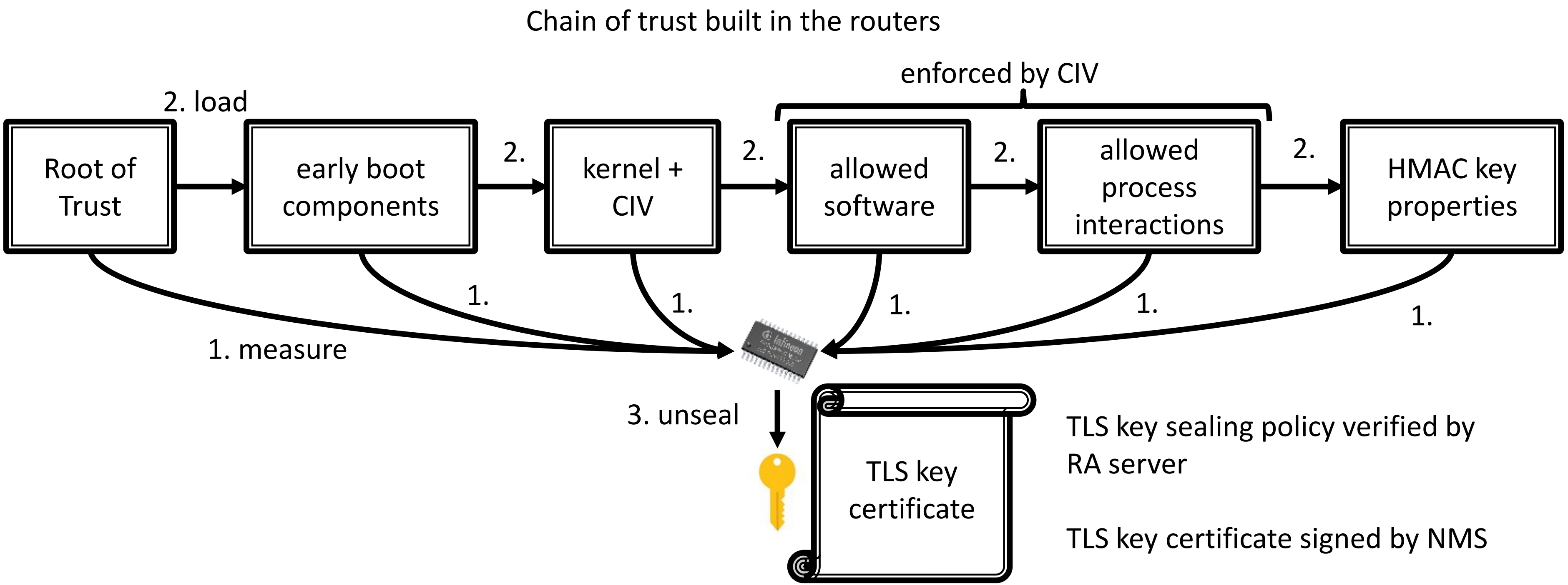
- Builds on top of existing software in the kernel security subsystem (IMA, EVM)

- Set of extensions for the Linux kernel to protect the integrity of a system for the entire lifecycle
  - IMA Digest Lists (load-time)
  - Infoflow LSM (run-time)
  - EVM with TPM key (offline)

- More complete protection/detection of the integrity of applications
  - Besides regular files, all process communication channels (socket, fifo, …) are considered

- Simplified integration of remote attestation into existing products
  - Remote attestation implicitly done during the establishment of a trusted channel

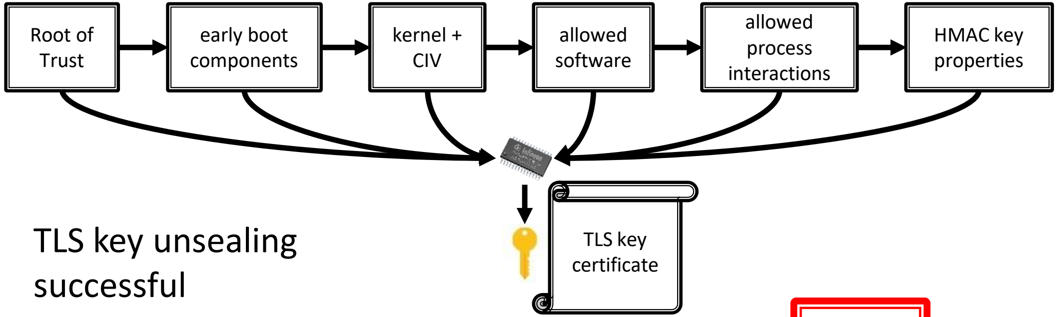
CIV and TPM



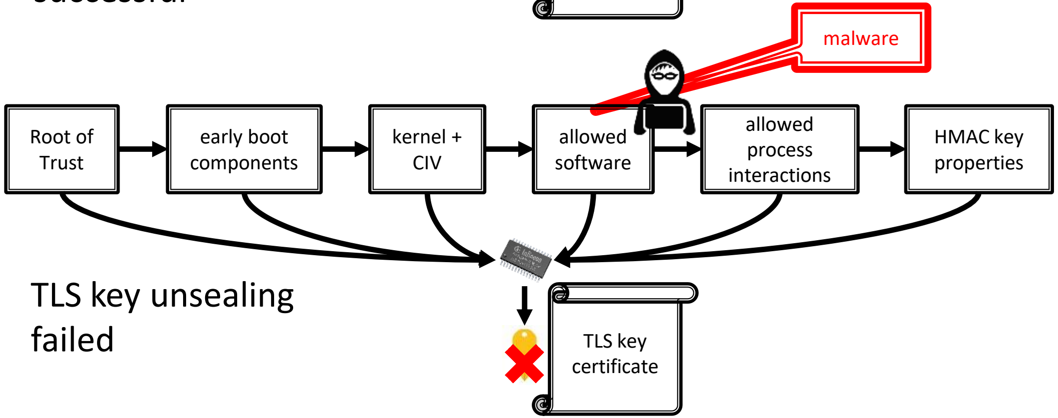
TLS key sealing policy verified by RA server

TLS key certificate signed by NMS

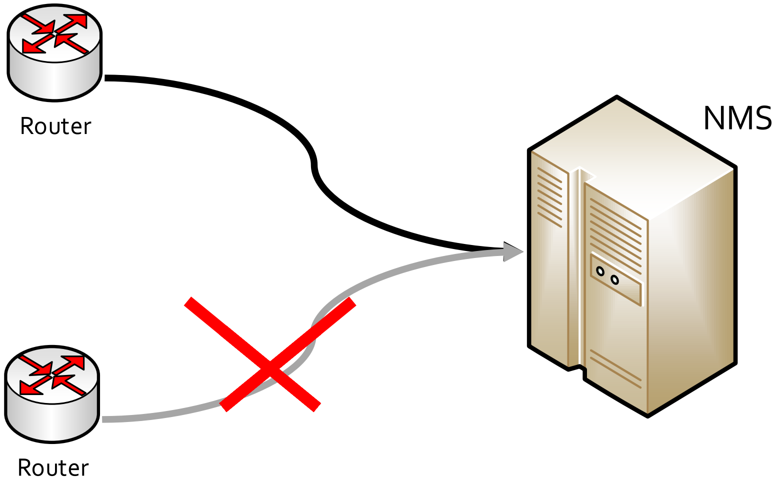
Implicit Remote Attestation



TLS key unsealing successful



TLS key unsealing failed

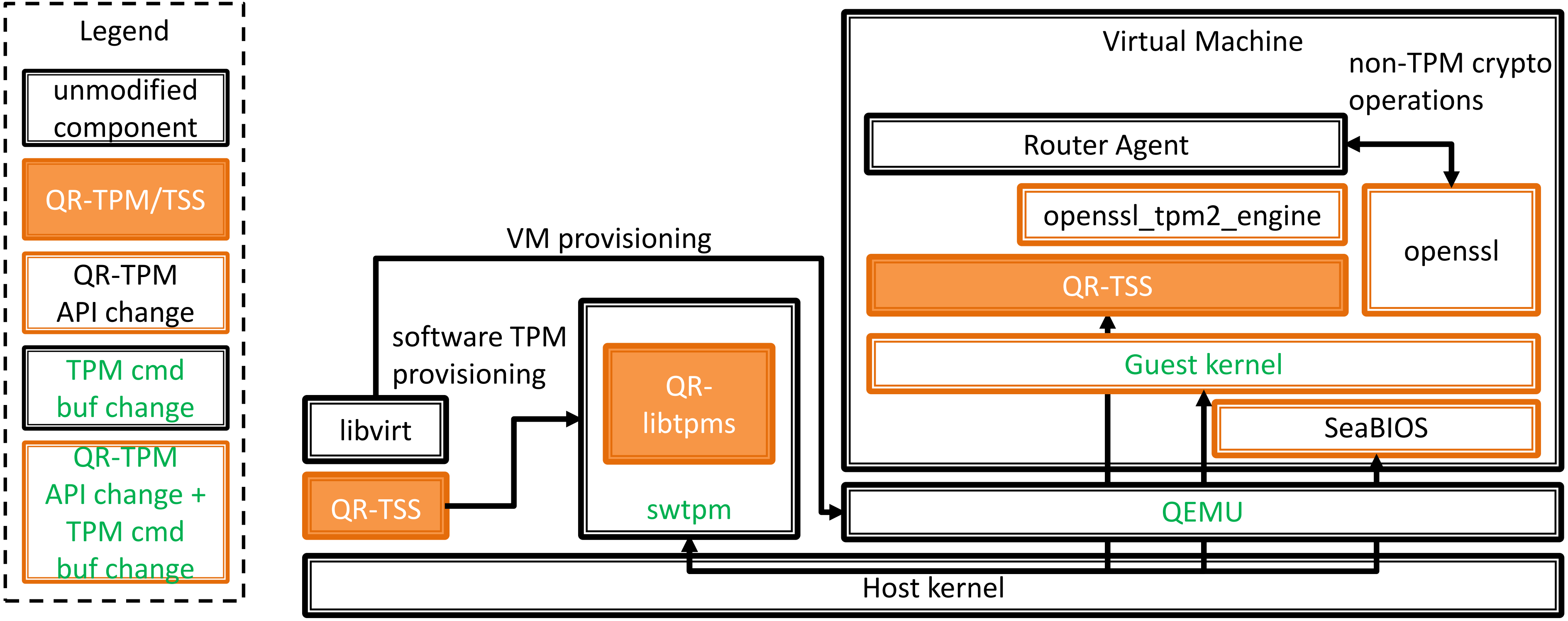


A compromised router cannot establish a TLS connection with the certificate issued by the NMS

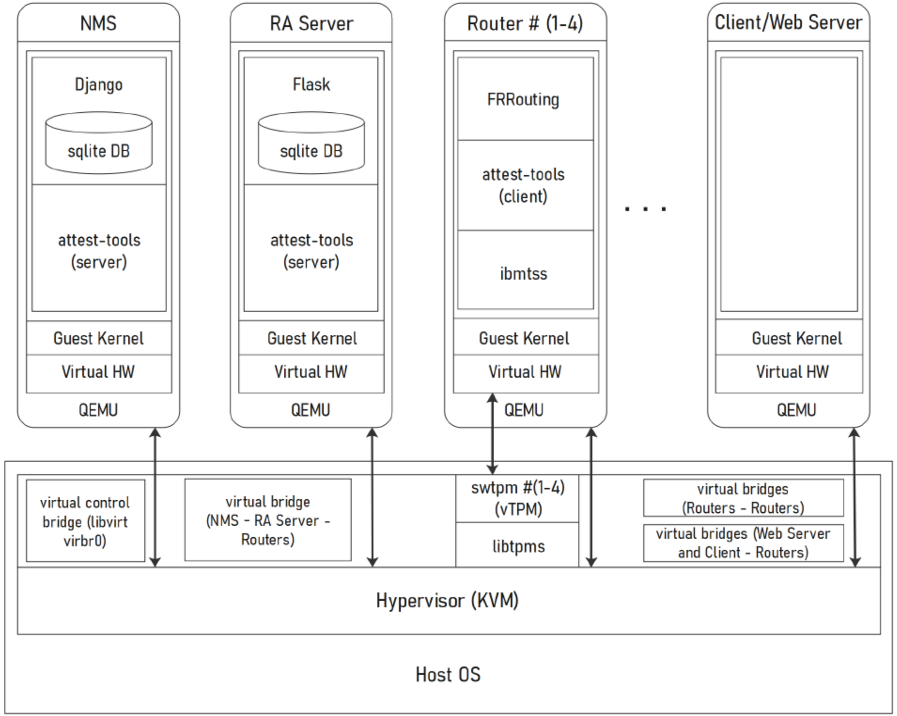
Secure Zero Touch Provisioning

- Routers are admitted to the network if they have a valid certificate
- Routers are configured to get a valid certificate at the first boot and their current configuration must match the one defined by the Network Administrator
- During operation, any change from the verified configuration causes the unsealing of TLS key in the TPM to fail
- If a malicious Network Operator tries to subvert a router before or after the router gets a certificate, the NMS will notice it (enrollment or TLS connection fails)

Integration of Software TPM

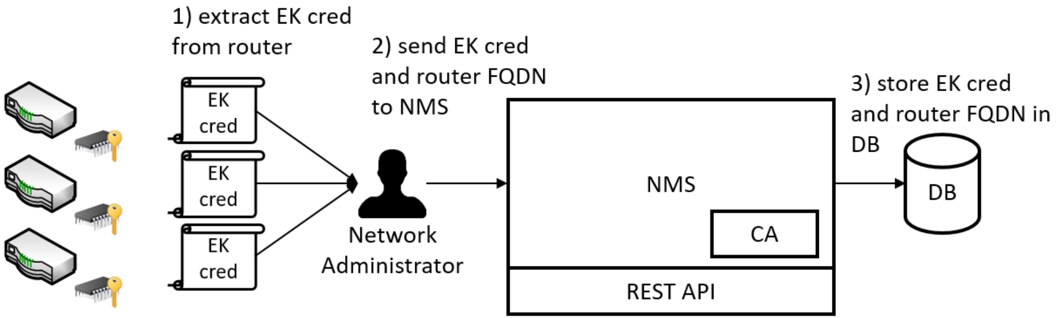


Demo Setup


NMS
Django
sqlite DB
attest-tools (server)
Guest Kernel
Virtual HW
QEMU
RA Server
Flask
sqlite DB
attest-tools (server)
Guest Kernel
Virtual HW
QEMU
Router # (1-4)
FRRouting
attest-tools (client)
ibmtss
Guest Kernel
Virtual HW
QEMU
. . .
Client/Web Server
Guest Kernel
Virtual HW
QEMU
virtual control bridge (libvirt virbr0)
virtual bridge (NMS - RA Server - Routers)
swtpm #(1-4) (vTPM)
libtpms
virtual bridges (Routers - Routers)
virtual bridges (Web Server and Client - Routers)
Hypervisor (KVM)
Host OS

User Story Demo: HWDU.NA.1

As a Network Administrator, I want to enrol the router with the NMS so that it is accepted in the network infrastructure.



User Story Demo: HWDU.NA.2

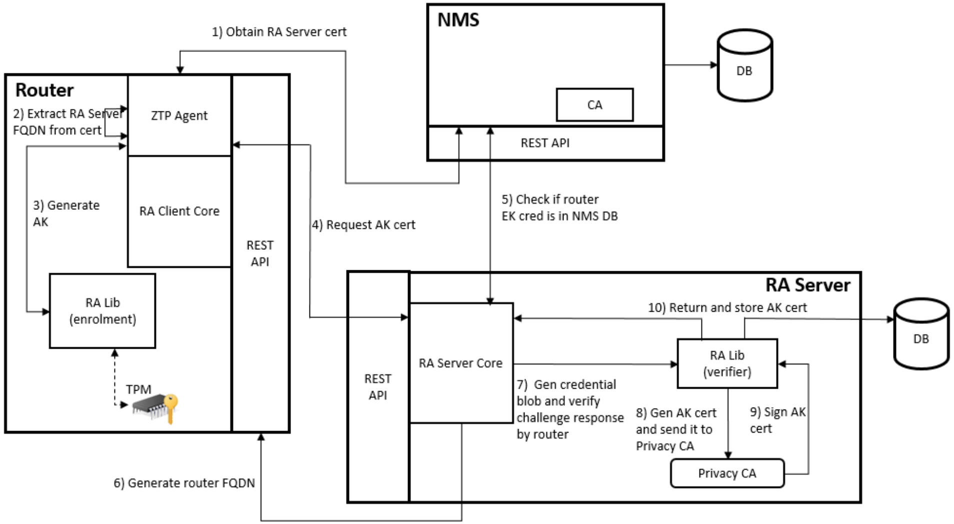
As a Network Administrator I want to define a trusted routing policy on the NMS so that the traffic is processed according to the trust states of routers.

Integrity Status	Routing Table Metric
good	10
unknown	20
bad	30
offline	40

Mapping table with pre-defined values

User Story Demo: HWDU.NO.1 – Establish Trust in TPM

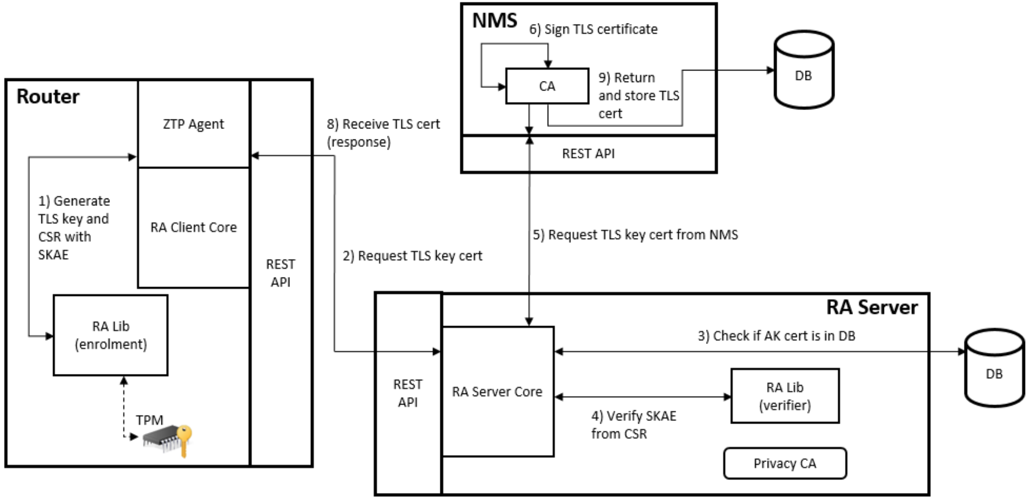
The Network Operator connects the router to the network and is able to verify the device integrity based on a whitelist.*



* List of reference fingerprint values for files in the router image, signed by the vendor

User Story Demo: HWDU.NO.1 – Certify Router Config

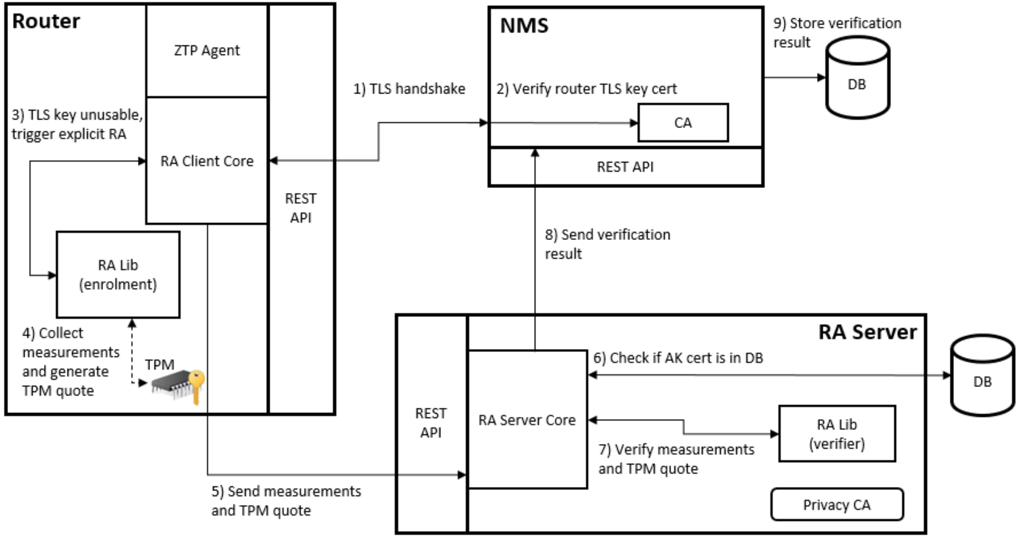
The Network Operator connects the router to the network and is able to verify the device integrity based on a whitelist.*



* List of reference fingerprint values for files in the router image, signed by the vendor

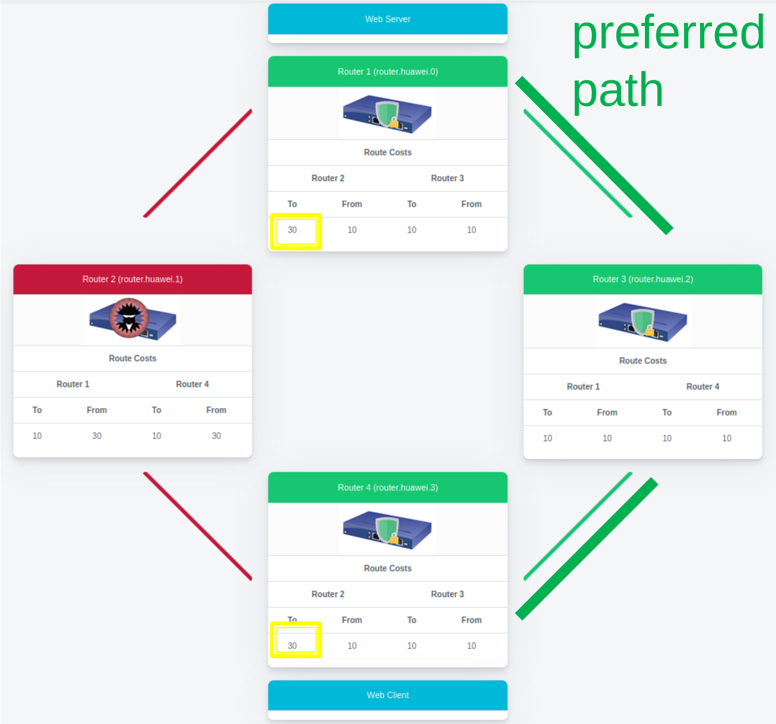
User Story Demo: HWDU.NA.4

As a Network Administrator I want to monitor the overall trust state of the network infrastructure.

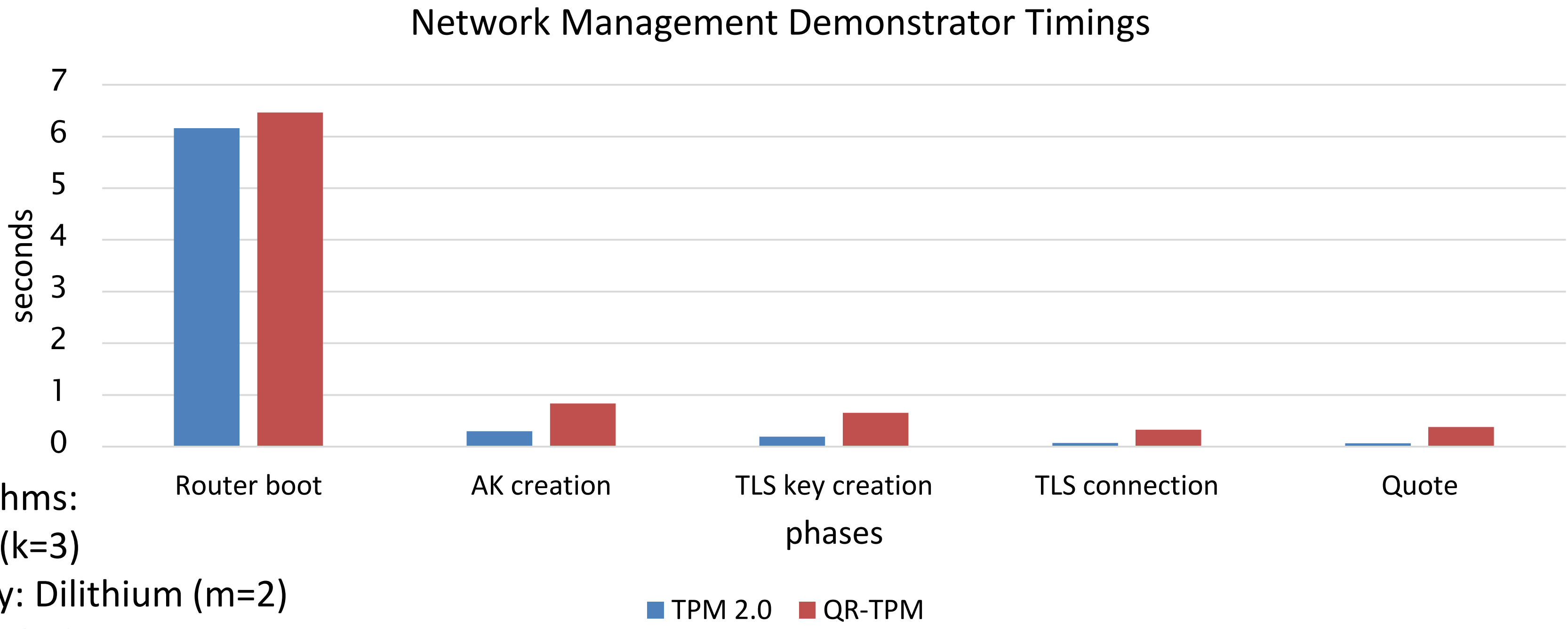


User Story Demo: HWDU.NA.3

As a Network Administrator I want to enforce the trusted routing policy in the network to reduce the risk of traffic leaking by untrusted routers.



TPM Performance Evaluation



QR algorithms:
EK: Kyber (k=3)
AK/TLS key: Dilithium (m=2)
Hash: SHA-256

TPM Command Timings

Most of TPM commands are ~10 times slower with QR-TPM

TPM 2.0 Command	TPM 2.0 Timings (TSS)	FutureTPM Command	FutureTPM Timings (TSS)
Router Boot	**6.159**		**6.466**
TPM2_StartAuthSession	N/A		N/A
TPM2_PolicyPCR (SHA1)	N/A	TPM2_PolicyPCR (SHA256)	N/A
TPM2_Unseal	N/A		N/A
AK Creation	**0.300**		**0.834**
TPM2_Create (AK, rsa 2048)	0.004779	TPM2_Create (AK, dilithium mode=2)	0.031657
TPM2_CreatePrimary (EK, rsa 2048)	0.011244	TPM2_CreatePrimary (EK, kyber security=3)	0.020212
TPM2_Load (AK, rsa 2048)	0.002805	TPM2_Load (AK, dilithium mode=2)	0.030117
TPM2_ActivateCredential	0.002394		0.018827
TLS Key Creation	**0.194**		**0.655**
TPM2_PCR_Read (SHA1)	0.000789	TPM2_PCR_Read (SHA256)	0.013633
TPM2_Create (TLS, rsa 2048)	0.004865	TPM2_Create (TLS, dilithium mode=2)	0.032031
TPM2_Load (TLS, rsa 2048)	0.002942	TPM2_Load (TLS, dilithium mode=2)	0.030333
TPM2_Load (AK, rsa 2048)	0.002779	TPM2_Load (AK, dilithium mode=2)	0.030129
TPM2_Certify	0.002279		0.023121
TPM2_StartAuthSession (SRK used as salt key)	0.001963		0.018708
TPM2_PolicyPCR (SHA1)	0.000601	TPM2_PolicyPCR (SHA256)	0.013880
TPM2_RSA_Decrypt	0.003242	TPM2_Sign	0.022728
TLS Connection	**0.073**		**0.331**
TPM2_ReadPublic (SRK, rsa 2048)	0.002401	TPM2_ReadPublic (SRK, kyber security=3)	0.018779
TPM2_StartAuthSession(SRK used as salt key)	0.002068		0.018585
TPM2_Load (TLS, rsa 2048)	0.003677	TPM2_Load (TLS, dilithium mode=2)	0.030866
TPM2_PolicyPCR (SHA1)	0.000623	TPM2_PolicyPCR (SHA256)	0.013606
TPM2_RSA_Decrypt	0.003241	TPM2_Sign	0.022806
Quote	**0.066**		**0.381**
TPM2_Load (AK, rsa 2048)	0.003126	TPM2_Load (AK, dilithium mode=2)	0.029669
TPM2_Quote	0.002785		0.022542

Network Performance Evaluation

In a sample experiment, 90.8% of the packets were successfully diverted away from the compromised router

In a real scenario (e.g. a Zoom call of 31 minutes*), the percentage becomes 99.92%

Wireshark · Capture File Properties · capture_compromised.pcap

Details

File

Name:	/home/ivan/simple_ra/capture_compromised.pcap
Length:	81 kB
Hash (SHA256):	e0e3d281126e446e0a359ffc3d46b67bb5aa80193b11e321c95d9b8fed174188
Hash (RIPEMD160):	f6e5442be4d37093d322c9ecf9157c37f74ea20f
Hash (SHA1):	cd279ba6d80b4ac634e02eb4b3a0a2c42eeddd8d
Format:	Wireshark/tcpdump/... - pcap
Encapsulation:	Linux cooked-mode capture
Snapshot length:	262144

Time

First packet:	2020-10-26 16:02:51
Last packet:	2020-10-26 16:02:58
Elapsed:	00:00:07

Capture

Hardware:	Unknown
OS:	Unknown
Application:	Unknown

Interfaces

Interface	Dropped packets	Capture filter	Link type	Packet size limit
Unknown	Unknown	Unknown	Linux cooked-mode capture	262144 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	704	639 (90.8%)	—
Time span, s	7.222	6.572	—
Average pps	97.5	97.2	—
Average packet size, B	100	100	—
Bytes	70400	63900 (90.8%)	0
Average bytes/s	9,747	9,723	—
Average bits/s	77 k	77 k	—

*https://skillscouter.com/video-conferencing-statistics/

Quantitative KPI

Id	Metric	Target Value	Acceptance criteria	(M)andatory / (G)ood to Have / (O)ptional	Measured by M36
1	Amount of routers whose integrity is monitored by NMS	100%	100%	M	With TPM2.0: 100% With FutureTPM: 100%
2	Amount of routers hiding their integrity status	0%	0%	M	With TPM2.0: 0% With FutureTPM: 0%
3	Amount of detected integrity attacks on routers	80% (with integrity models)	60% (standard IMA)	M	With TPM2.0: 80% With FutureTPM: 80%
4	Amount of traffic diverted to alternative paths when a router is compromised	75%	55%	G	With TPM2.0: 90.8% With FutureTPM: 90.8%
5	Amount of files whose integrity can be verified	100% (with integrity models)	99% (standard IMA)	G M	With TPM2.0: 100% With FutureTPM: 100%

Reasonably pessimistic estimation, in a real scenario measured values are better

Qualitative KPI

Id	Metric	Target Value	(M)andatory / (G)ood to Have / (O)ptional	Measured by M36
1	Traffic routing based on router trust state	Supported	M	With TPM2.0: Supported With FutureTPM: Supported
2	Trusted channels between NMS and each router in the network	Supported	M	With TPM2.0: Supported With FutureTPM: Supported
3	Device authentication key for trusted channel protected by TPM	Supported	M	With TPM2.0: Supported With FutureTPM: Supported
4	Integrity protection of router configuration data using a TPM key	Supported	M	With TPM2.0: Supported With FutureTPM: Supported

Conclusions

- Migration from TPM 2.0 to QR-TPM is feasible and is fully compatible with the system integrity use cases of trusted computing, with reasonable performance impact
- TPM and trusted computing are a foundation for system security in network infrastructures and new trust-based use cases can be built on top of them
- Quantum resistance must be implemented across the entire trusted computing stack (from TPM firmware to crypto libraries and TLS)

FutureTPM Grant Agreement No. 779391

"The FutureTPM project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779391."

If you need further information, please contact the coordinator:

TECHNIKON Forschungs- und Planungsgesellschaft mbH
Burgplatz 3a, 9500 Villach, AUSTRIA
Tel: +43 4242 233 55 Fax: +43 4242 233 55 77
E-Mail: coordination@futuretpm.eu