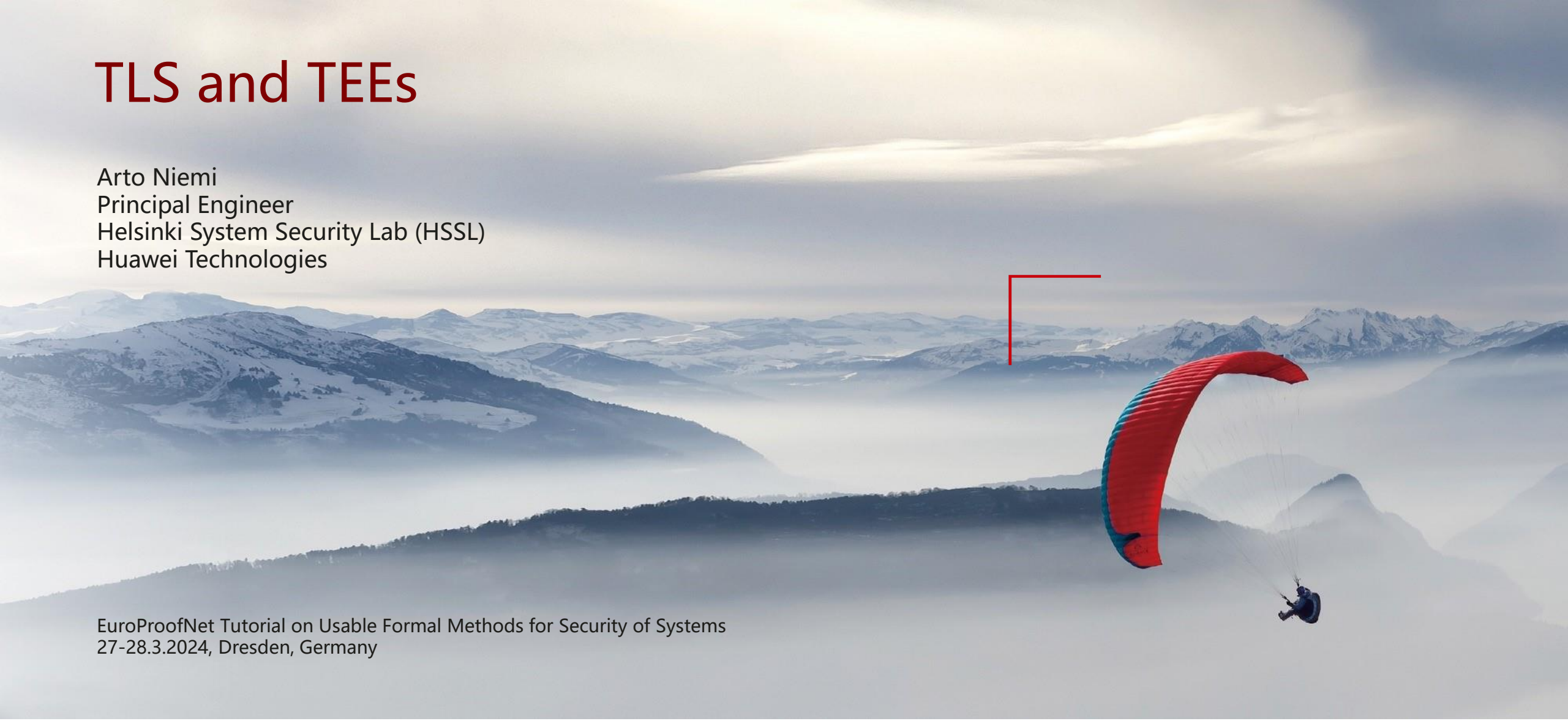
TLS and TEEs

Arto Niemi
Principal Engineer
Helsinki System Security Lab (HSSL)
Huawei Technologies



EuroProofNet Tutorial on Usable Formal Methods for Security of Systems
27-28.3.2024, Dresden, Germany

HUAWEI

Overview of the talk

- TLS: Transport Layer Security
  - Widely-used for secure communication (HTTPS)
- TEE: Trusted Execution Environment
  - Widely-used to protect keys (smartphones) and workloads (cloud)
- TLS and TEE: a happy marriage (?)
  - Benefit 1: TEE makes TLS more secure
  - Benefit 2: TLS helps TEE to communicate securely
  - History, integration and standardization

Helsinki System Security Lab (HSSL)

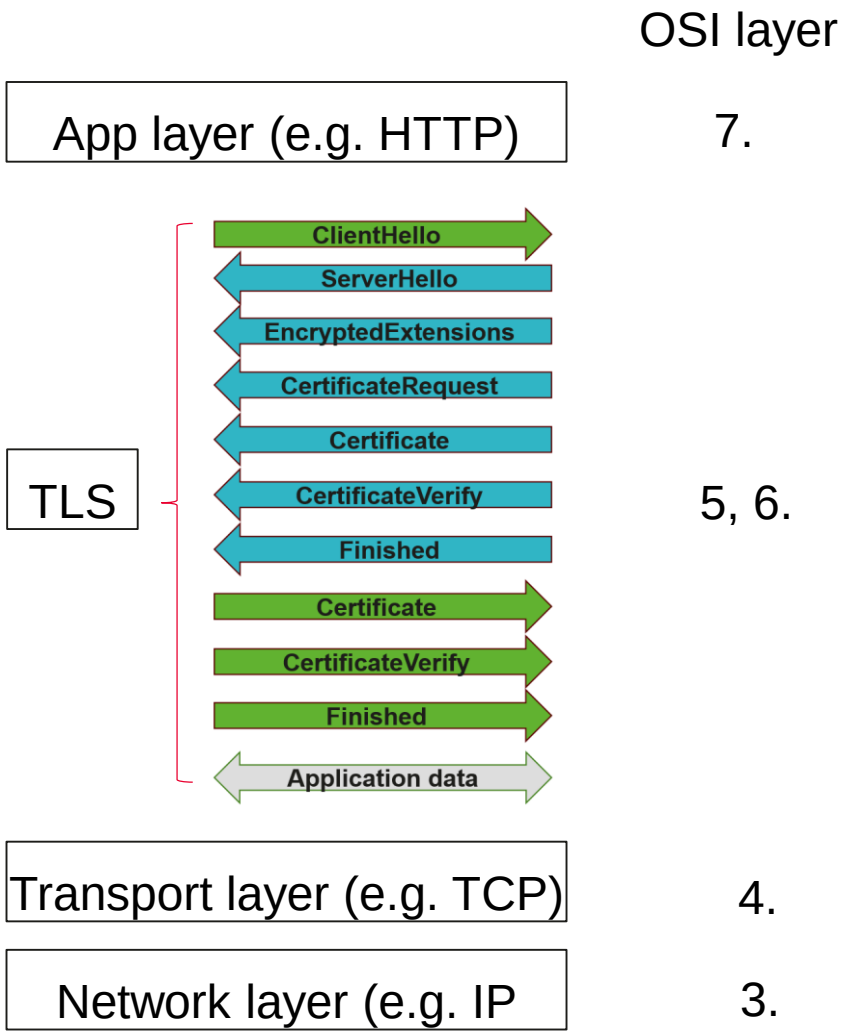
- Part of Huawei Finland Research Center (FiRC)
  - Around 25 employees (250+ in whole FiRC)
  - Focuses on platform & system security
  - Lots of former Nokia Research Center employees

- Significant TEE expertise
  - Contributions to TrustedCore/iTrustee, Huawei Unified Keystore (HUKS)
  - Representation in GlobalPlatform (TES Platforms & Services WGs), Linux CCC

- My earlier (pre-2020) background was in TLS & crypto implementation
  - Found TEE use cases for TLS: device key provisioning, enclave migration, device health attestation

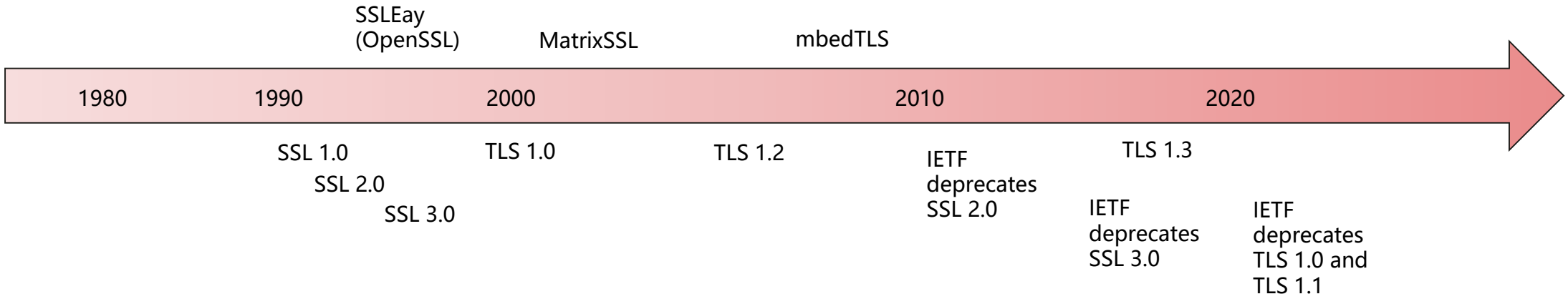
Transport Layer Security (TLS)

- World's most widely used secure communication protocol
  - > The "S" in HTTPS
  - > Client-server
  - > Conceptually: layers 5 & 6 in OSI model
- Establishes a secure channel
  - > Confidentiality, integrity for data
  - > Authentication of endpoints
  - > Replay protection, key confirmation, etc.
- TLS 1.3: 1.5 round-trips on top of TCP/IP
  - > UDP-based variant: DTLS



HUAWEI

Timeline of TLS



- SSL 1.0
  - Netscape internal protocol
  - Flawed design: no message integrity or sequence numbers
- SSL 2.0
  - Only 1 certificate per endpoint
  - Insecure MD5 for integrity protection
- SSL 3.0
  - First version developed with support from crypto experts (e.g. Taher ElGamal)
  - POODLE attack

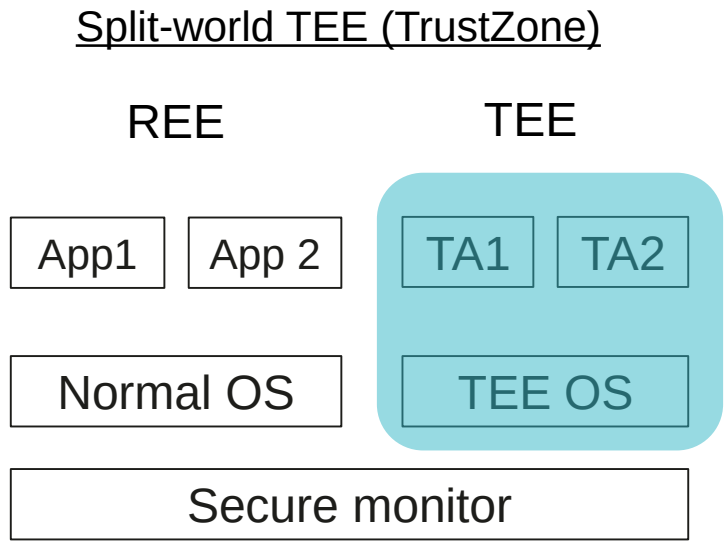
- TLS 1.0
  - IETF takes over from Netscape
  - Basically the same protocol as SSL 3.0
  - Attacks against RC4 ciphersuites
  - BEAST attack against CBC padding
- TLS 1.1
- TLS 1.2
  - Renegotiation attack
  - Heartbleed (OpenSSL)
  - CRIME attack against compression
  - ROBOT attack against PKCS #1.5

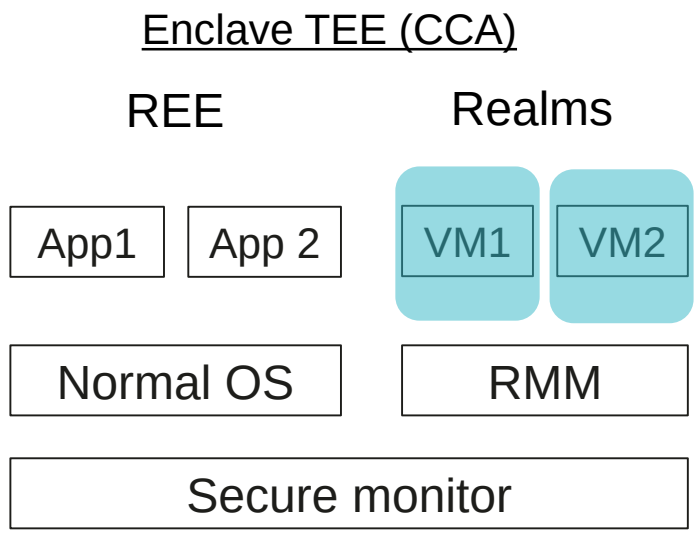
- TLS 1.3
  - Simplified handshake
  - Well-designed key schedule with strong binding of secrets to particular handshake
  - Less round-trips
  - Only allow secure crypto
  - First formally verified version
- Widely-used implementations
  - OpenSSL for high-end devices
  - mbedTLS for embedded use cases

Trusted Execution Environment (TEE)

- Provides hardware-based support for:
  - Isolated execution
  - Protected storage (sealing)
  - Ability to convince remote verifiers (remote attestation)

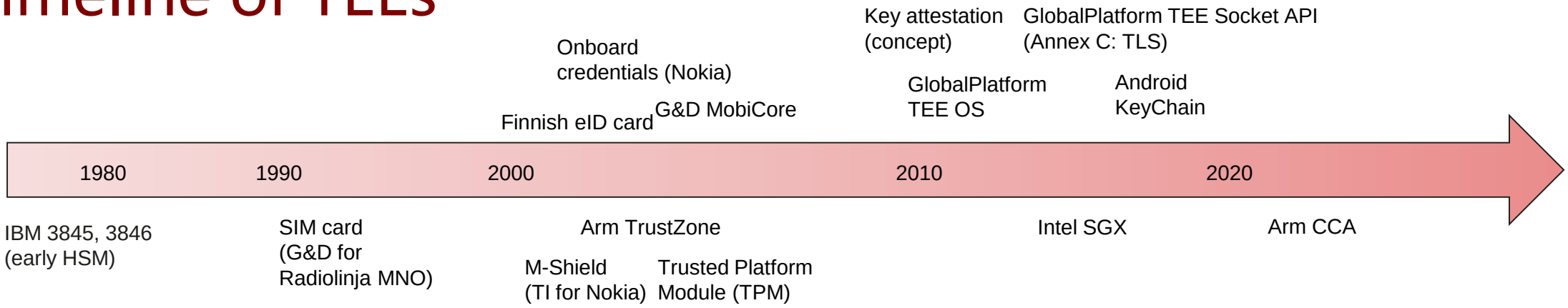
- Three varieties:
  - 1. External secure co-processor (e.g. HSM, CryptoCard, SIM card)
  - 2. Embedded secure co-processor (e.g. TPM, SEP, eSIM)
  - 3. Processor secure environment:
    - Split-world TEE (TrustZone)
    - Enclave TEE (SGX, CCA)





Asokan: Hardware-assisted Trusted Execution Environments. Look back, look ahead

Timeline of TEEs



- Hardware Security Module (HSM)
  - Protects and operates keys
  - IBM 3845 and 3846 crypto devices (1977)
  - Clears secrets when tampered with (FIPS 140)
- Subscriber Identity Module
  - Protects IMSI, keys, small programs
  - Invented by G&D for Finnish MNO Radiolinja
- Trusted Platform Module (TPM)
  - Cheaper than a HSM
  - Measured boot, sealing
  - Required by Windows 10+

- M-Shield
  - First generic TEE for mobile devices
  - Developed by TI in collaboration with Nokia
- Arm TrustZone
  - First widely deployed TEE hardware
  - Additional secure mode in the main CPU
- G&D MobiCore
  - First widely deployed TrustZone-based TEE
- Intel SGX
  - First widely deployed PSE for PCs
  - Significantly inspired academic research on TEEs

- Key attestation
  - Proof that key cannot leave TEE
  - Concept: Kostiainen et al. (2010)
  - Practical deployment: Android Key Store + keymaster (2016)
- GlobalPlatform
  - First standard for TEE OS
  - Widely followed by mobile TEE vendors
  - TMF and OTrP: way for vendors to provision keys into TEE apps
  - Socket API: way for apps in TEE to communicate with Internet servers
    - v.1.1 added TLS 1.3
    - v.1.2 added attested TLS
    - v.1.x add server-side TLS ?

- Confidential Computing
  - “TEEs in the cloud”
  - Trend towards VM-level TEEs
  - TLS: de-facto standard for communicating with cloud-based TEEs

Gunn et al.: Hardware platform security for mobile devices, 2022

TLS does not verify endpoint security

"Using encryption on the Internet is the equivalent of arranging an armored car to deliver credit information from someone living in a cardboard box to someone living on a park bench"

- Gene Spafford

Client

Server

TLS does not verify endpoint security

TLS

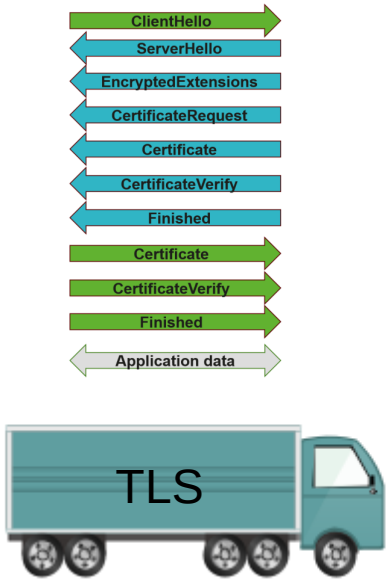
"Using ~~encryption~~ on the Internet is the equivalent of arranging an armored car to deliver ~~credit information~~ from someone living in a cardboard box to someone living on a park bench"

secrets, confidential

workloads

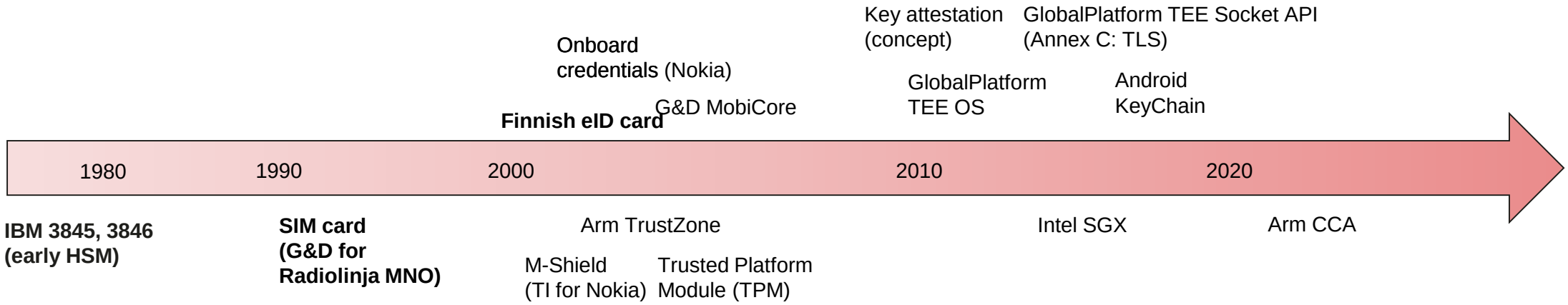
Gene Spafford

Client



Server

External secure co-processors



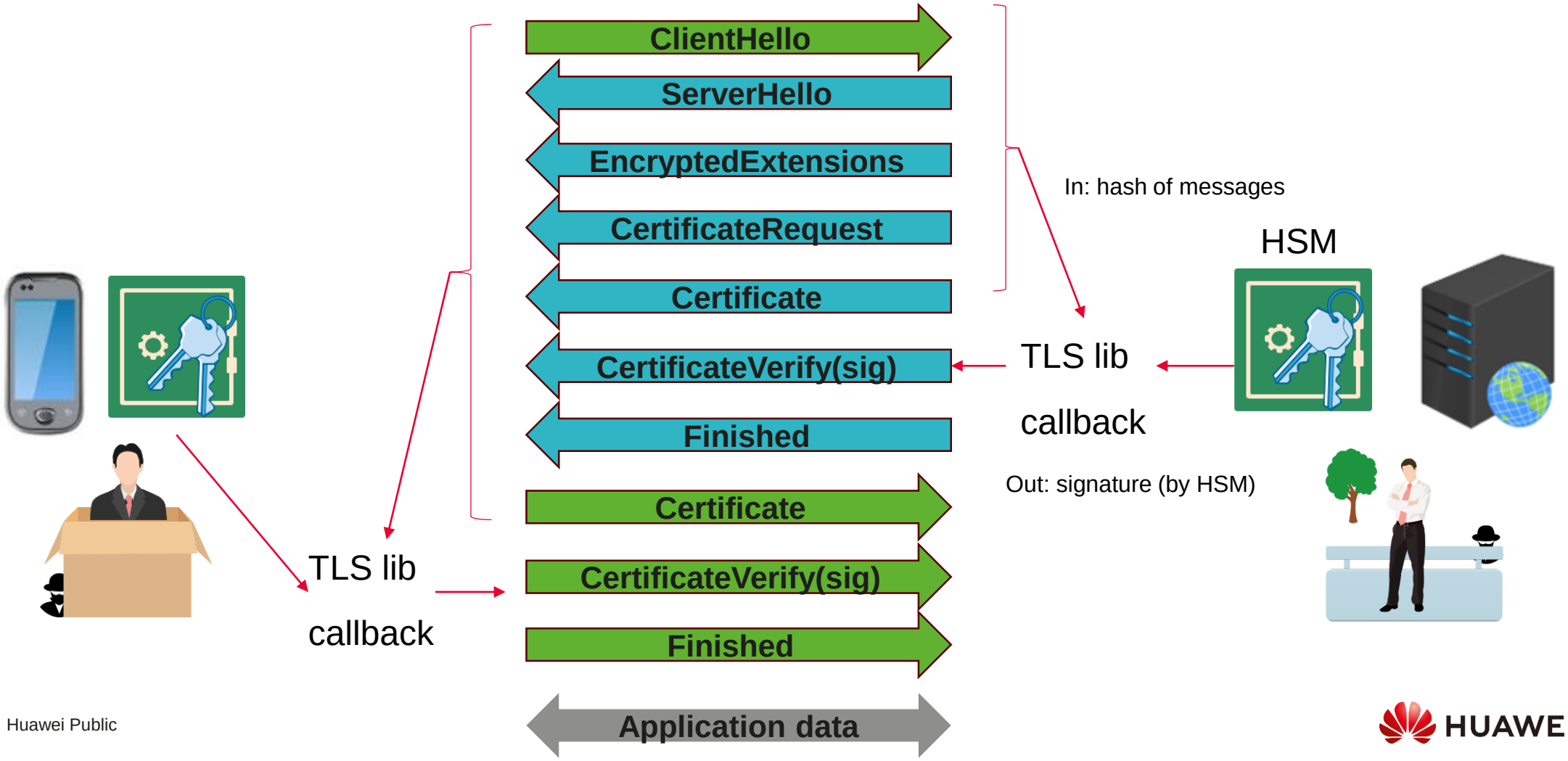
- Hardware Security Module (HSM)
  - Protects and operates cryptographic keys
  - IBM 3845 and 3846 crypto devices (1977) were among the first
  - Early usage: finance (ATMs), military communication
  - Used especially to protect certificate signing keys (e.g. in EMV key management)
  - Clears secrets when tampered with (FIPS 140)

- Subscriber Identity Module
  - Protects IMSI, symmetric keys and small programs
  - Invented by G&D for Finnish MNO Radiolinja
  - Typically not combined with TLS (c.f. GP SCP)

- EU electronic identity (eID) cards
  - Protects identity, TLS client authentication private key
  - Finnish and Estonian eID cards were first to have privkey + cert

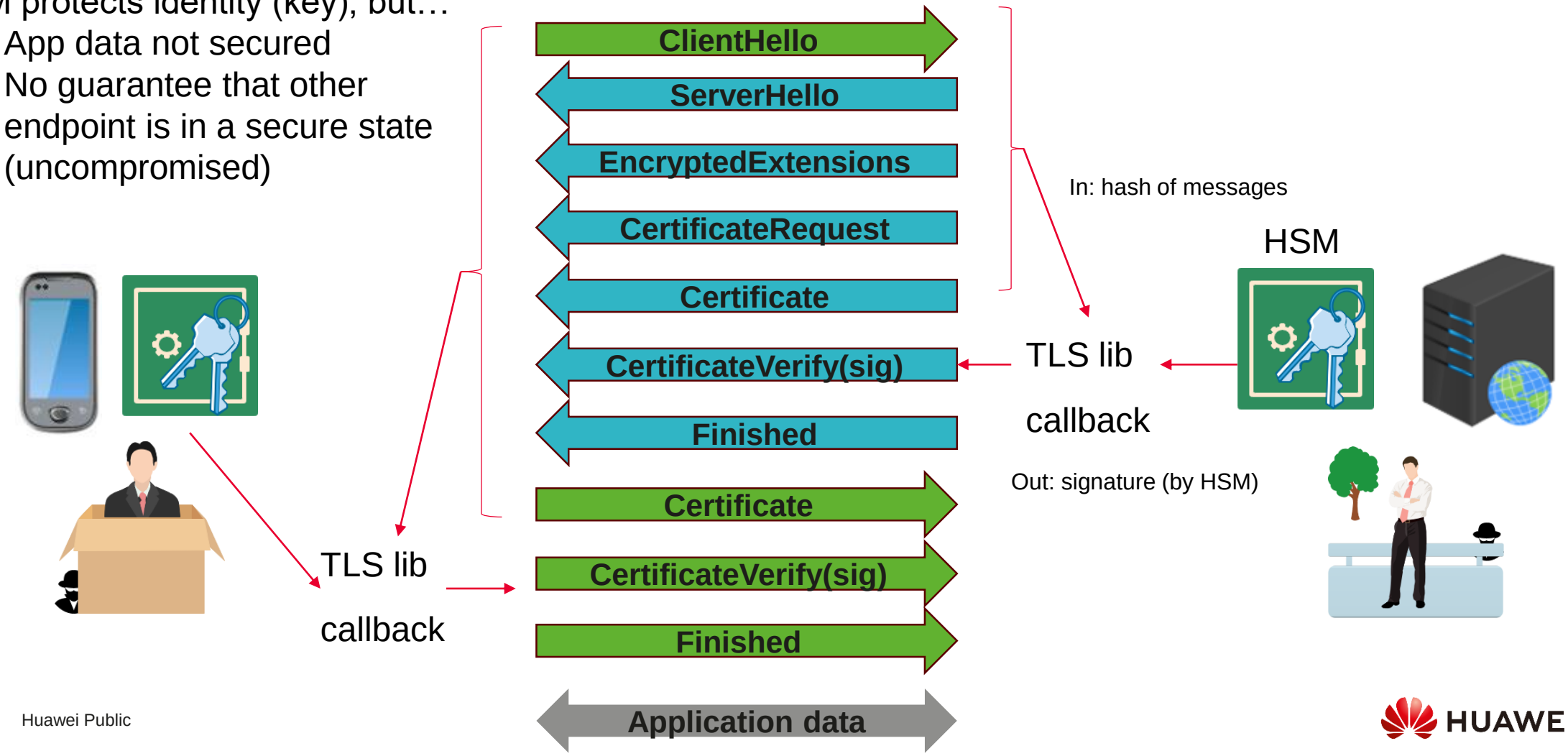
Securing TLS with HSM

HSM protects endpoint authentication private key and signs the hash of messages (for CertificateVerify)

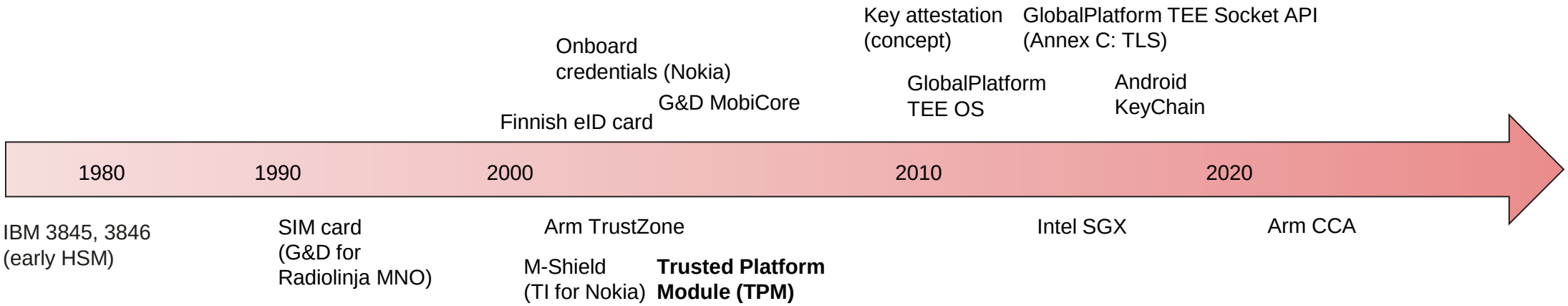


Securing TLS with HSM

- HSM protects identity (key), but…
  - App data not secured
  - No guarantee that other endpoint is in a secure state (uncompromised)



Embedded secure co-processor



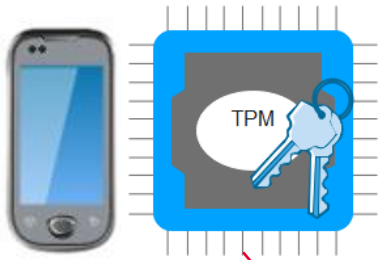
- Trusted Platform Module (TPM)
  - > Cheaper than a HSM
  - > Protects keys
  - > Can attest the TLS endpoint (measured boot)
  - > Required by Windows 10+

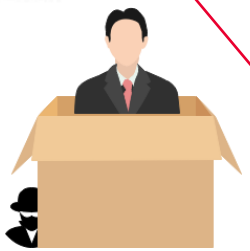
- Apple Secure Enclave Processor (SEP)
- Google Titan-M
- Microsoft Pluton

Securing TLS with TPM (one way)

Bootloader collects endpoint measurements (e.g. SW hashes), stores them in TPM

TPM attests to the measurement by signing them (TPM Quote)





TLS lib callback

ClientHello

ServerHello

EncryptedExtensions

CertificateRequest

Certificate (TPM Quote)

CertificateVerify(sig)

Finished

Certificate (TPM Quote)

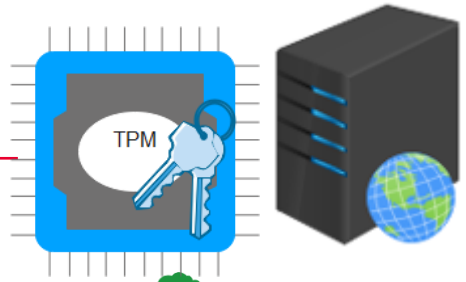
CertificateVerify(sig)

Finished

Application data

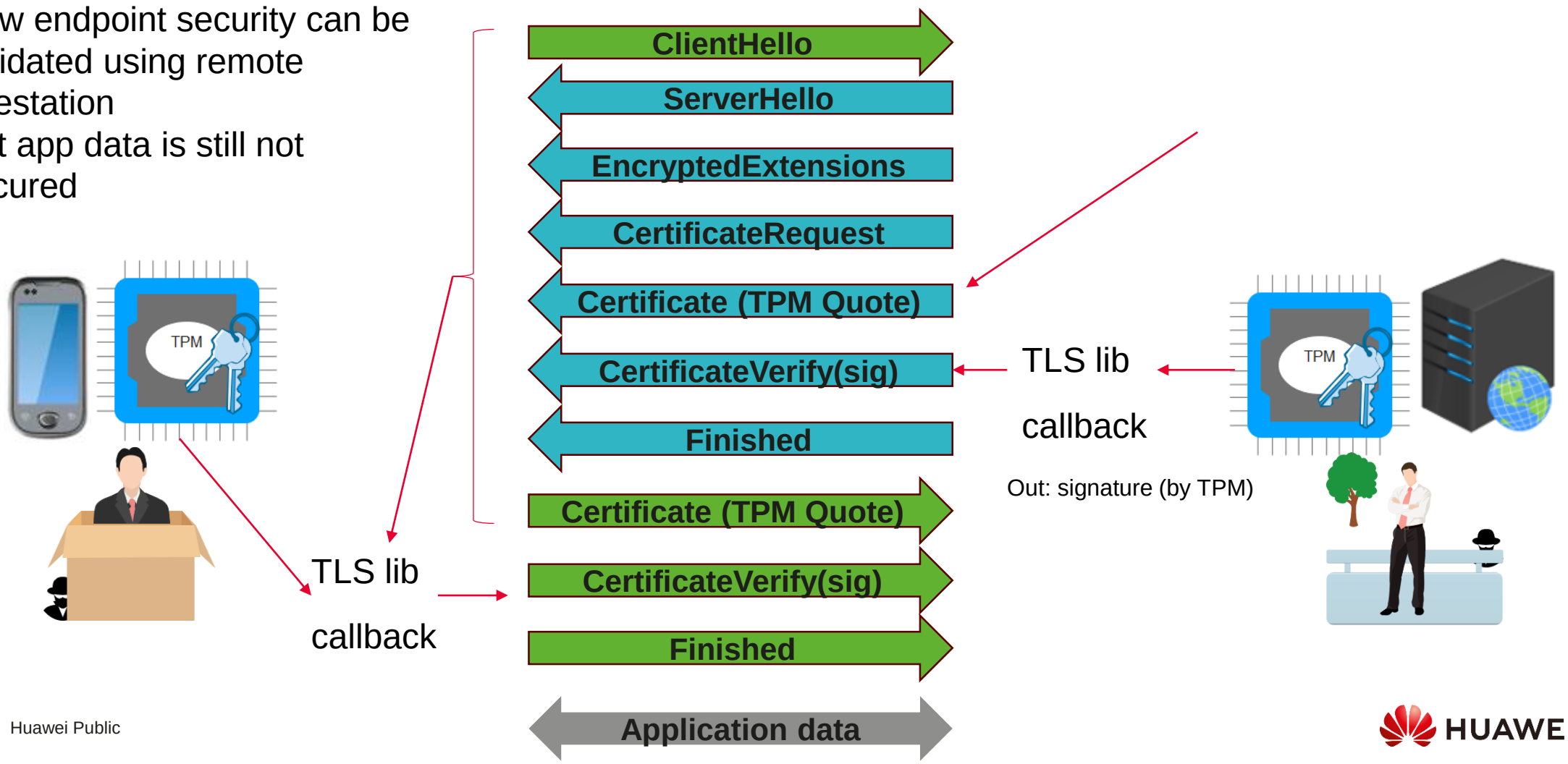
TLS lib callback

Out: signature (by TPM)

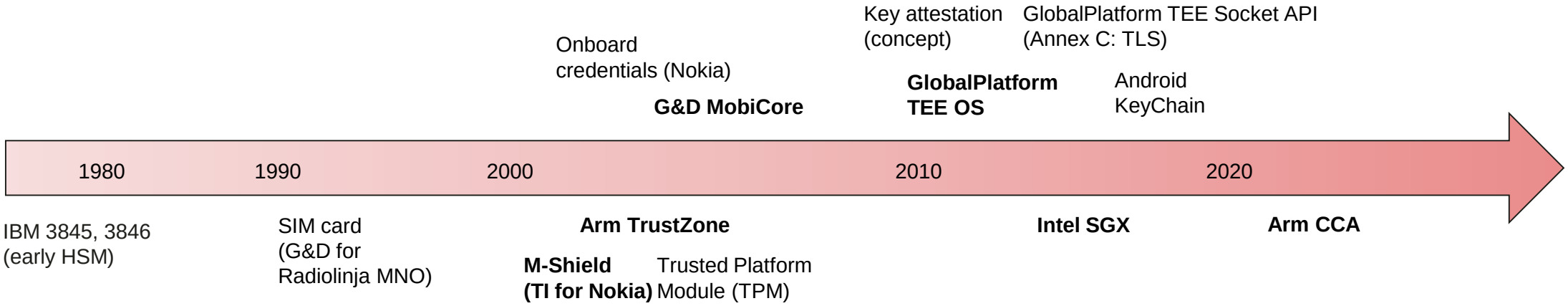


Securing TLS with TPM (one way)

- Now endpoint security can be validated using remote attestation
- But app data is still not secured



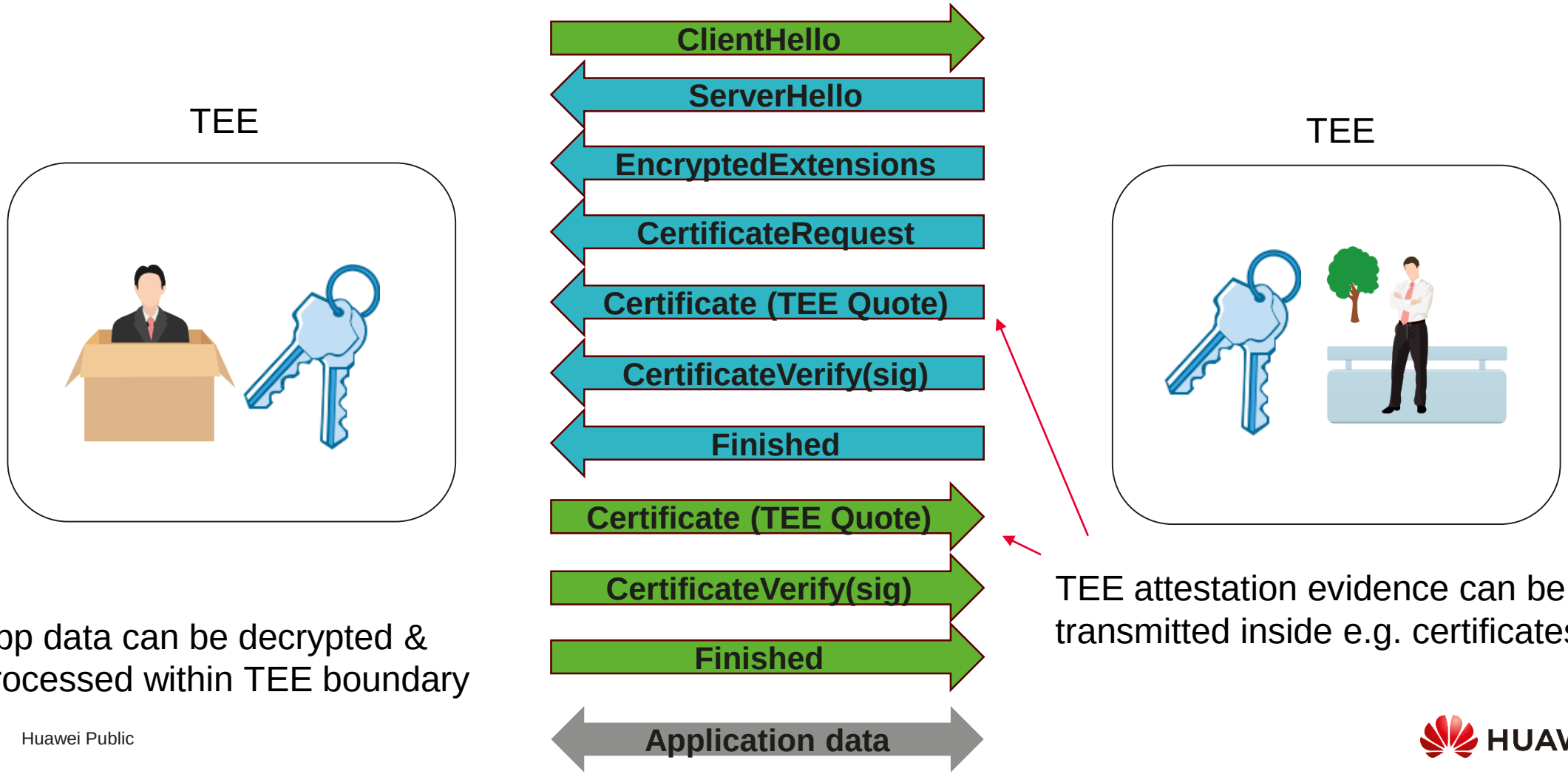
Processor secure environment (PSE)



- Initial motivation for PSEs:
  - Protect device identity (IMEI, subsidy lock)
  - Protect radio frequencies
  - Avoid increasing bill-of-materials
- M-Shield
  - First generic PSE hardware for mobile devices
  - Developed by TI in collaboration with Nokia
- Arm TrustZone
  - First widely deployed PSE hardware
  - Additional secure mode in the main CPU
- G&D MobiCore
  - First widely deployed PSE software (TEE OS)
- GlobalPlatform TEE specs
  - First standard for (TrustZone-like) TEEs
  - Defines APIs and security requirements
  - Widely followed by mobile phone TEEs
- Intel SGX
  - First widely deployed TEE for PCs
  - Significantly inspired academic research on TEEs
- Confidential Computing
  - “TEEs in the cloud”
  - Motivation: move sensitive processing from company premises to cloud
  - Trend towards VM-level TEEs (SEV-SNP, CCA)
  - TLS: de-facto standard for communication

Securing TLS with PSE

Whole TLS endpoint can be in the TEE



App data can be decrypted & processed within TEE boundary

TEE attestation evidence can be transmitted inside e.g. certificates

Recap

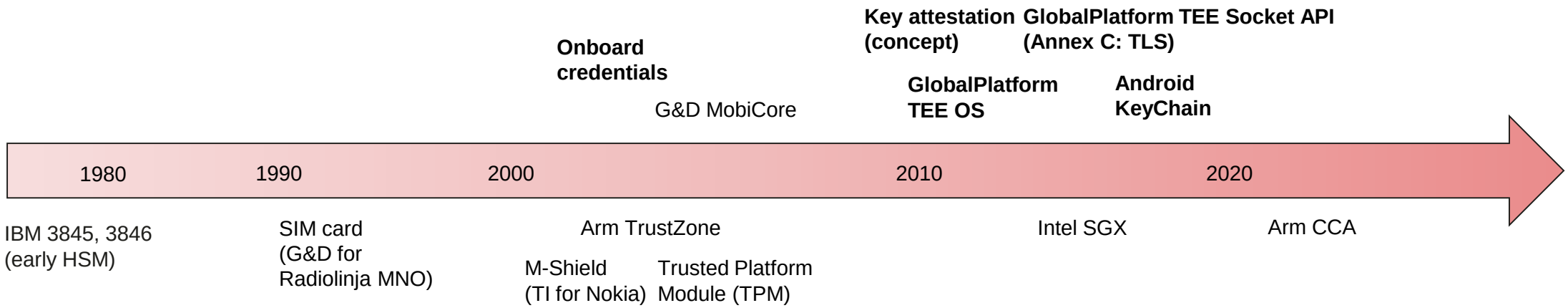
- Benefit 1: TEE makes TLS more secure
- Benefit 2: TLS helps TEE to communicate securely

1. TLS + HSM: secures endpoint identity
2. TLS + TPM: secure endpoint identity, boot-time integrity
3. TLS + PSE: secures whole TLS endpoint (keys, integrity, data)

TLS can help TEE to communicate

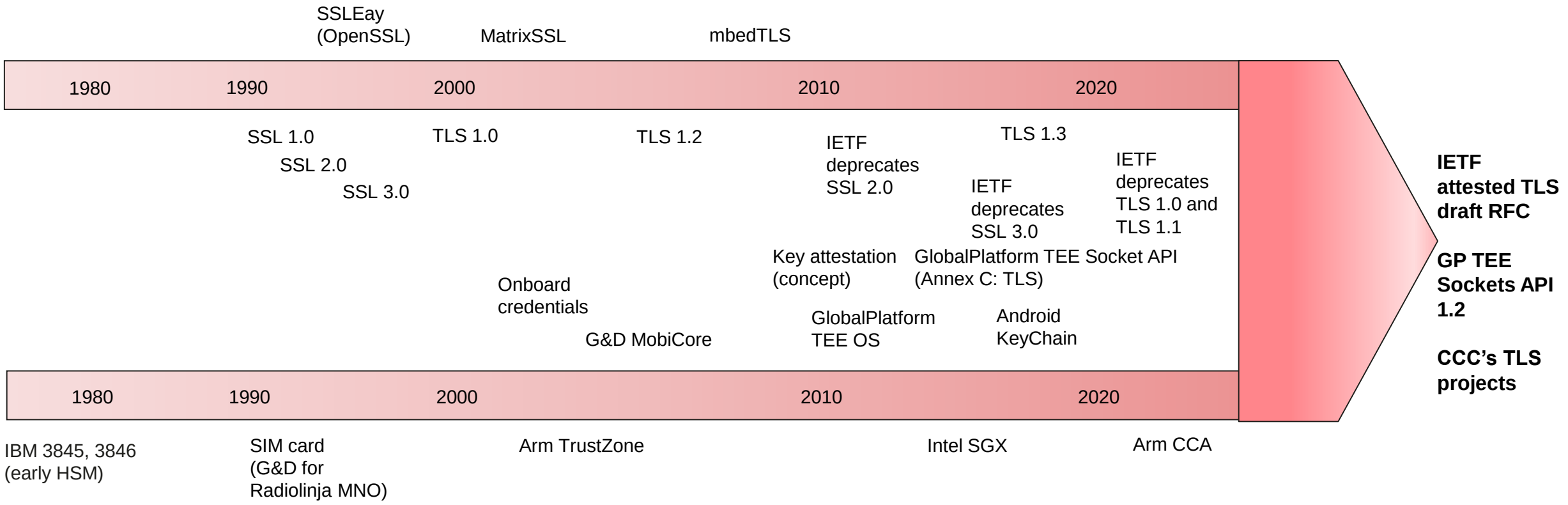
- Use case 1: provisioning
  - > Install / update TA, provision secrets (e.g. DRM keys)
  - > Provision confidential cloud workload (e.g. LLM)
- Use case 2: access to services
  - > TA in smartphone talks to backend (e.g. banking app)
  - > Migrate workloads between two TEEs
  - > Request services from cloud (e.g. query LLM)
- Need standards and interoperability

TEE attestation + communication timeline



- Key attestation
  - Proof that key cannot leave TEE
  - Concept: Kostiainen et al. (2010)
  - Practical deployment: Android KeyChain (2016)
- GlobalPlatform
  - First standard for TEE OS
  - Widely followed by mobile TEE vendors
  - TMF and OTrP: way for vendors to provision keys into TEE apps
  - Socket API: way for apps in TEE to communicate with Internet servers
    - v.1.1 added TLS 1.3
    - v.1.2 added attested TLS
    - v.1.x add server-side TLS ?
- Confidential Computing
  - “TEEs in the cloud”
  - Trend towards VM-level TEEs
  - TLS: de-facto standard for communicating with cloud-based TEEs
  - Harmonization via Linux CCC projects:

TLS + TEE timeline

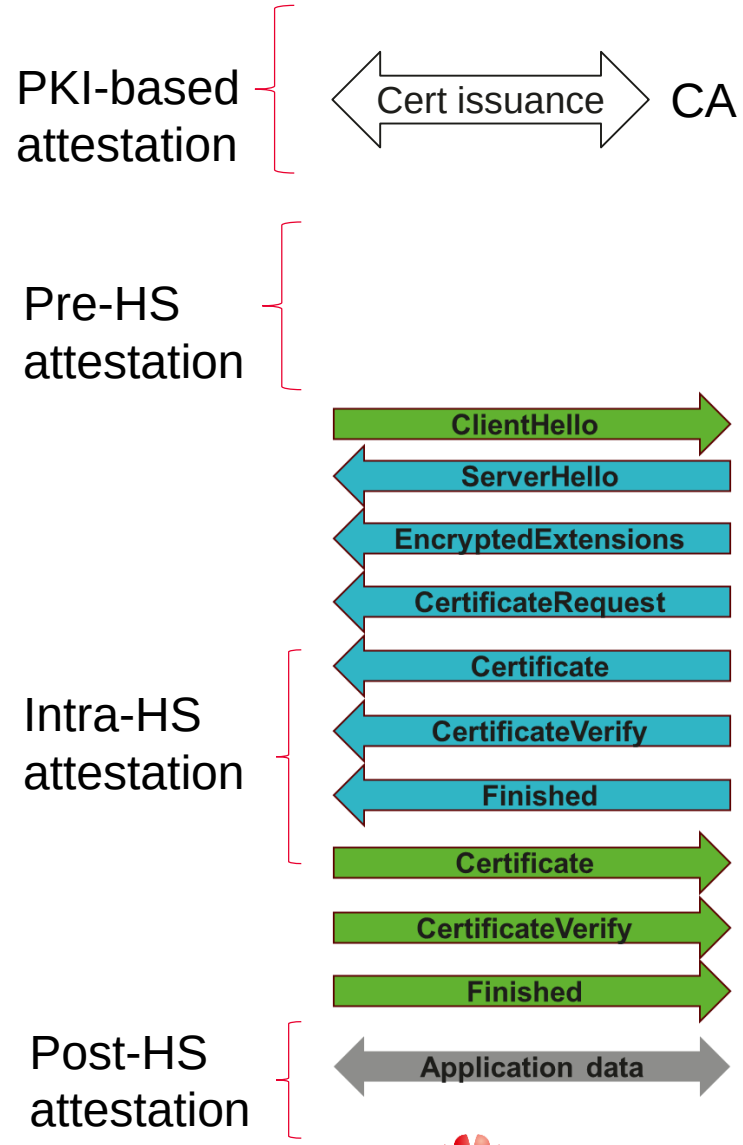


- IETF attested TLS draft RFC
  - Standard TLS extensions for performing remote attestation as part of handshake
- GlobalPlatform TEE Sockets API Annex C: TLS v.1.2
  - Standard API that TAs can call to use TLS for outgoing connections
  - Later: support for server-side TLS
- Linux CCC projects
  - Several attested TLS implementations
  - Veraison project for attestation evidence verification

HUAWEI

Integrating remote attestation and TLS

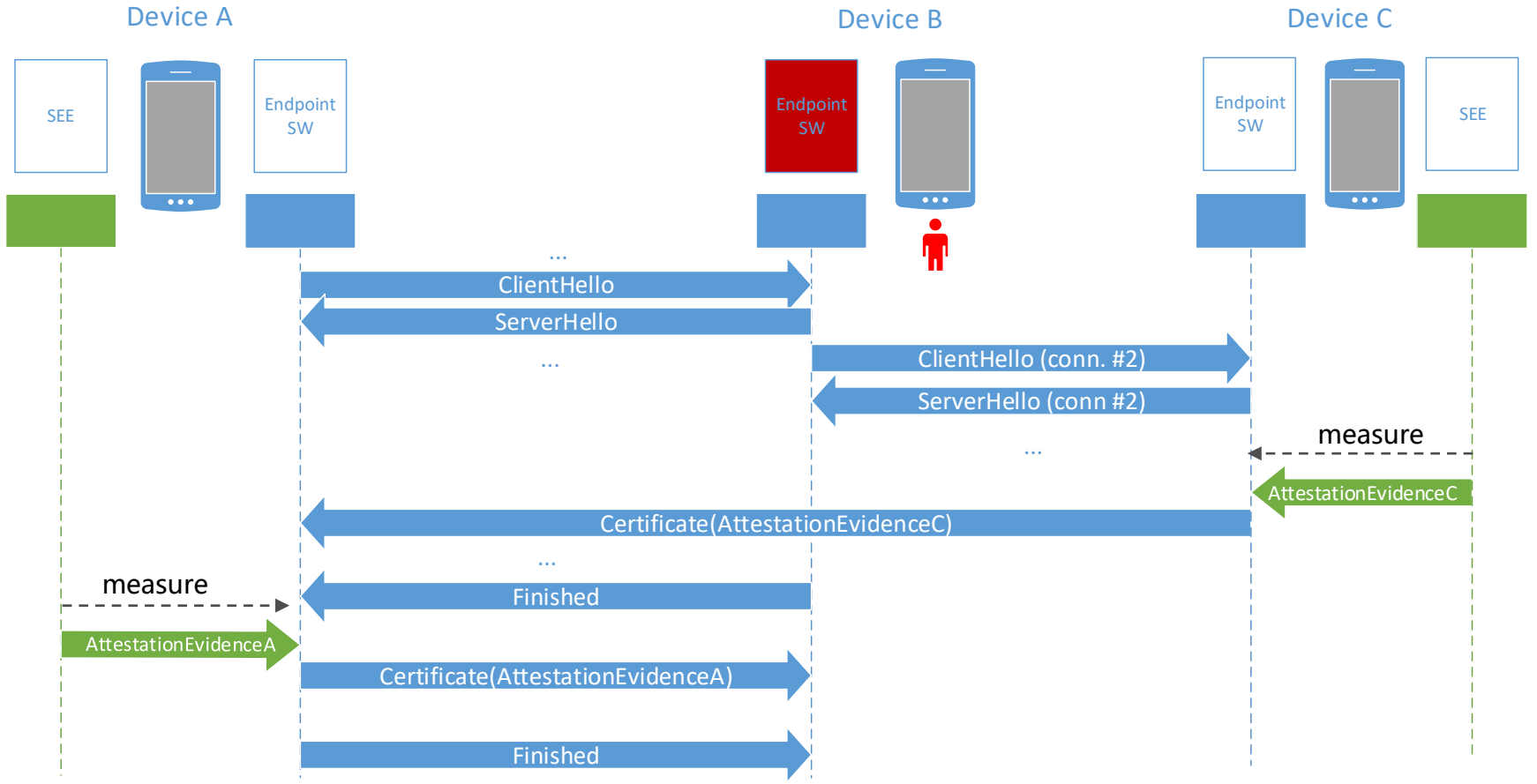
- Criteria:
  - Should not modify core protocol
  - Should be convenient to use with existing TLS libraries
  - Should be efficient (no extra round-trips)
  - Should be secure (channel binding)
- PKI-based attestation
  - CA appraises evidence during certificate issuance
  - Only attestation result, not evidence, is transmitted during HS
  - See later talk on attested CSR
- Pre-handshake
  - Evidence signed before HS (e.g. Intel RA-TLS)
- Post-handshake
  - Transmit evidence after HS (e.g. SCONE)
- Intra-handshake: sign & transmit evidence during HS
  - No extra roundtrips + strong channel binding
  - Works with existing TLS APIs (+extra configuration step)
  - Turns TLS into trusted channel (no intermediate phase)



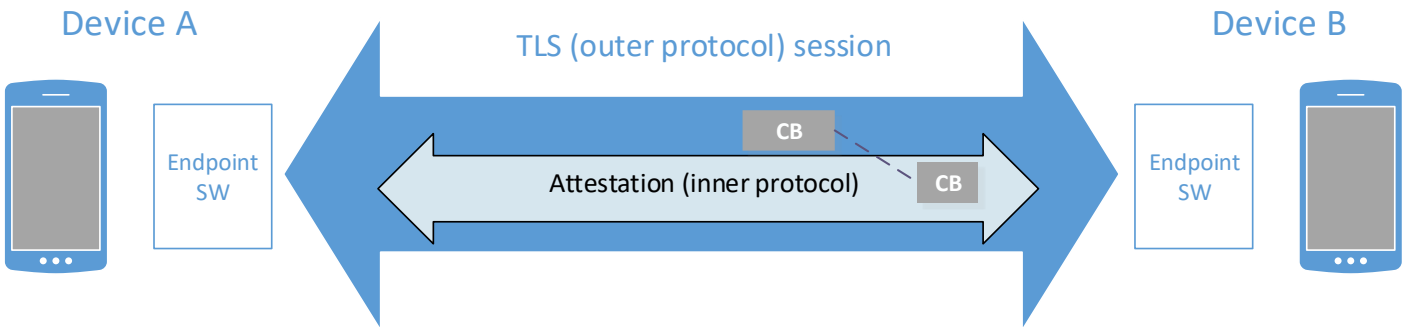
Niemi et al (2021), *Trusted Sockets Layer: A TLS 1.3 Based Trusted Channel Protocol*

Relay attack

- Attacker handshakes with an uncompromised device to get a valid-looking attestation evidence for his compromised device
- Possible because the attestation evidence was not bound to a specific TLS handshake or endpoint



Channel binding



- *Channel binding:*
  - Establishing that no man-in-the-middle exists between two end-points that have attested/authenticated each other in one (inner) protocol, but are using a secure channel provided by another, (outer) protocol
- *Channel bindings* (CB)
  - A unique identifier for a protocol session or endpoint
- *Explicit* channel binding
  - Endpoints compute CBs, transmit them over the wire
  - Endpoints check that self-computed CB matches received CB
- *Implicit* channel binding
  - CB of inner protocol is used in the key derivation of the outer protocol

Examples of attested TLS

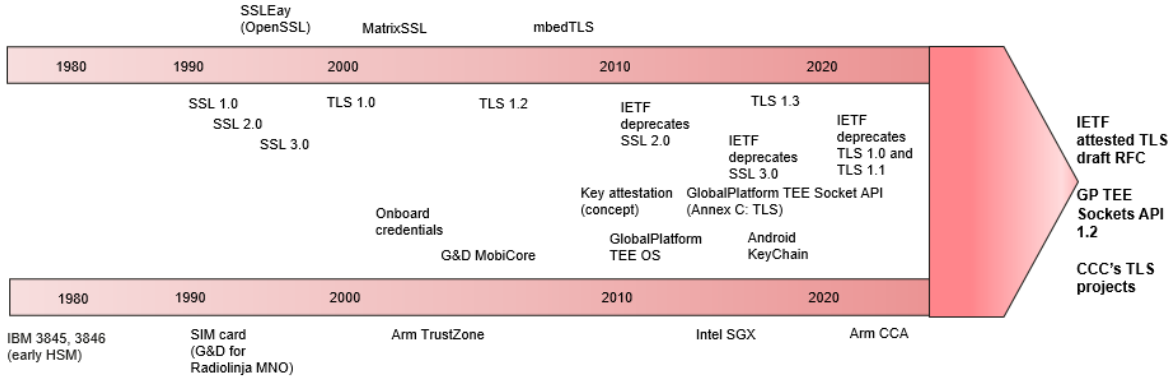
- Trusted Sockets Layer
  - Intra-handshake
  - Send evidence as an extension in X.509 endpoint authentication certificate
  - Use TLS-Exporter as channel bindings --> makes evidence valid only in a single handshake.

- IETF's attested TLS draft RFC
  - Intra-handshake
  - Allow multiple ways to send evidence
  - Extensions for requesting and transmitting evidence
  - Allow Web PKI and attestation certificates to co-exist
  - Optional channel bindings

- GlobalPlatform TEE Sockets API v1.2 Annex C
  - Intra- & post-handshake
  - Defines *APIs* for attested TLS
  - Now: write TA, use custom attested TLS protocol
  - After attested TLS is standardized: use it (without needing to change TA code) or keep using custom

Summary

- State of the TLS
  - > Initially developed for web browsers (online commerce), now omnipresent, used for most Internet traffic (HTTPS)
  - > Version 1.3 regarded secure, formally verified



- State of the TEE
  - > Categories: 1) external and 2) embedded co-processors, 3) processor secure environments (PSE)
  - > PSEs introduced in Nokia devices (IMEI, subsidy lock)
  - > Today very widely available:
    - Every smartphone has a PSE (TrustZone-based/GlobalPlatform-compliant)
    - Every PC has a TPM (for Windows/BitLocker)
    - Cloud providers deploying VM-granular TEEs (Confidential Computing): AMD SEV-SNP, Intel TDX

- TLS and TEE: a happy marriage? Yes!
  - > TEE makes TLS more secure
  - > TLS helps TEE to communicate securely
  - > Challenges:
    - > Integration of RA not yet a solved problem (formal verification will help)
    - > Lack of standard for attested TLS
    - > And what about PKI?
  - > Both now over 30 years old – it is about time!

Thank you! Any questions?

HUAWEI