Formal Specification and Verification of Attestation in Confidential Computing

Muhammad Usama Sardar

TU Dresden

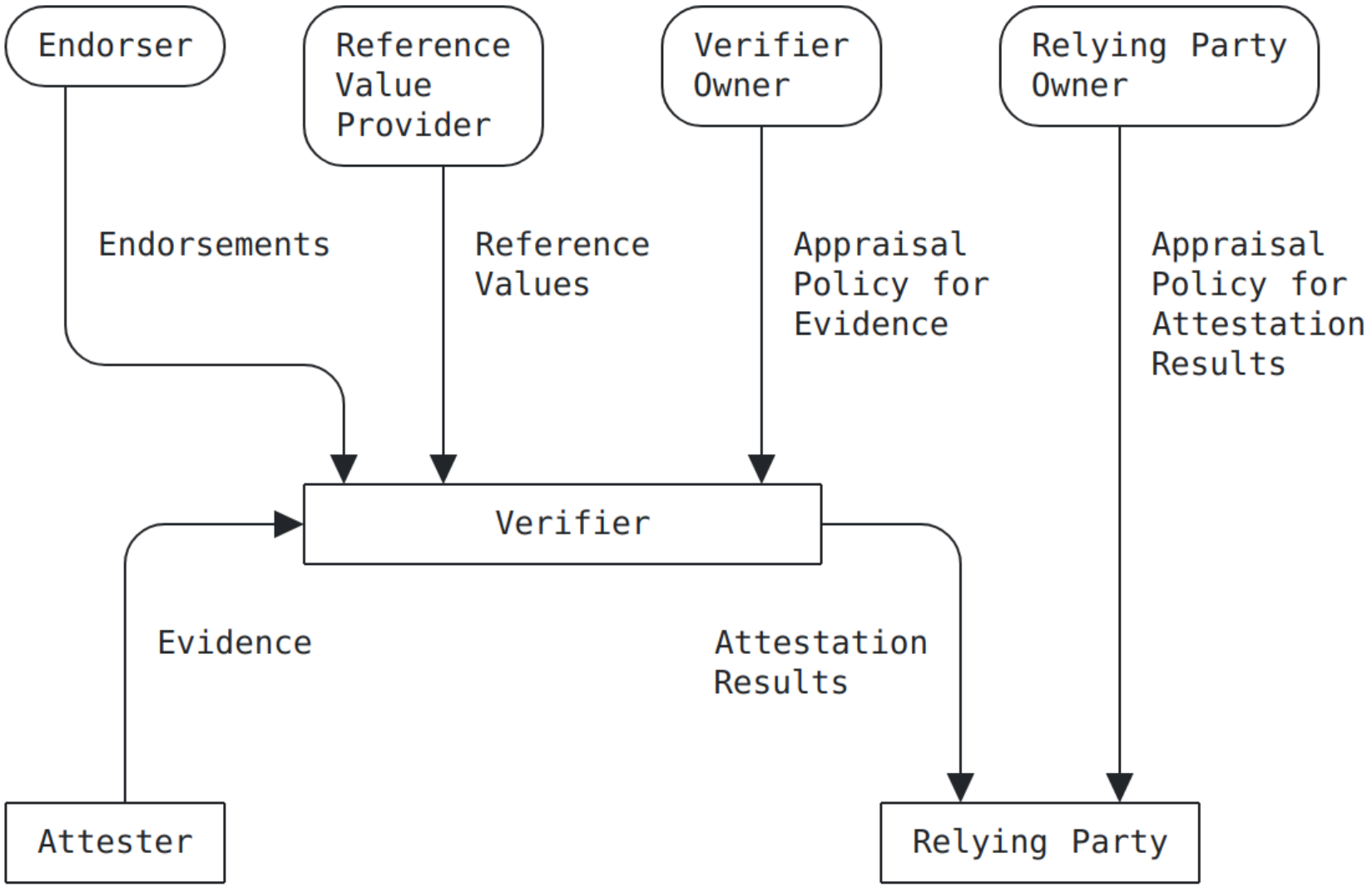
November 8, 2023

Funded by CPEC

Agenda

1. Motivation
2. Approach
3. Results
4. Summary

We all know RATS[1]



[1] Birkholz et al., *Remote ATtestation procedureS (RATS) Architecture*, 2023.

But is RATS sufficient for CC (e.g., SGX)?[2]

sgaxe.com

Signing Your Own Quotes

We understand that remote attestion can be very tricky to pass. However, since we already done all the hard work of getting genuine attestation keys, we decided to help you out by developing a Twitter bot that passes SGX attestation for you. Our bot provides Attestation as a Service (AaaS), which allows you to get your own quotes signed with the keys we extracted using SGAxe. This way you can pass attestation without even owning an SGX machine. If you want to make use of our service, you can send a tweet to our bot @SGAxe_AaaS. If you'll tweet it, we'll sign it!

@SGAxe_AaaS

[2] www.sgaxe.com

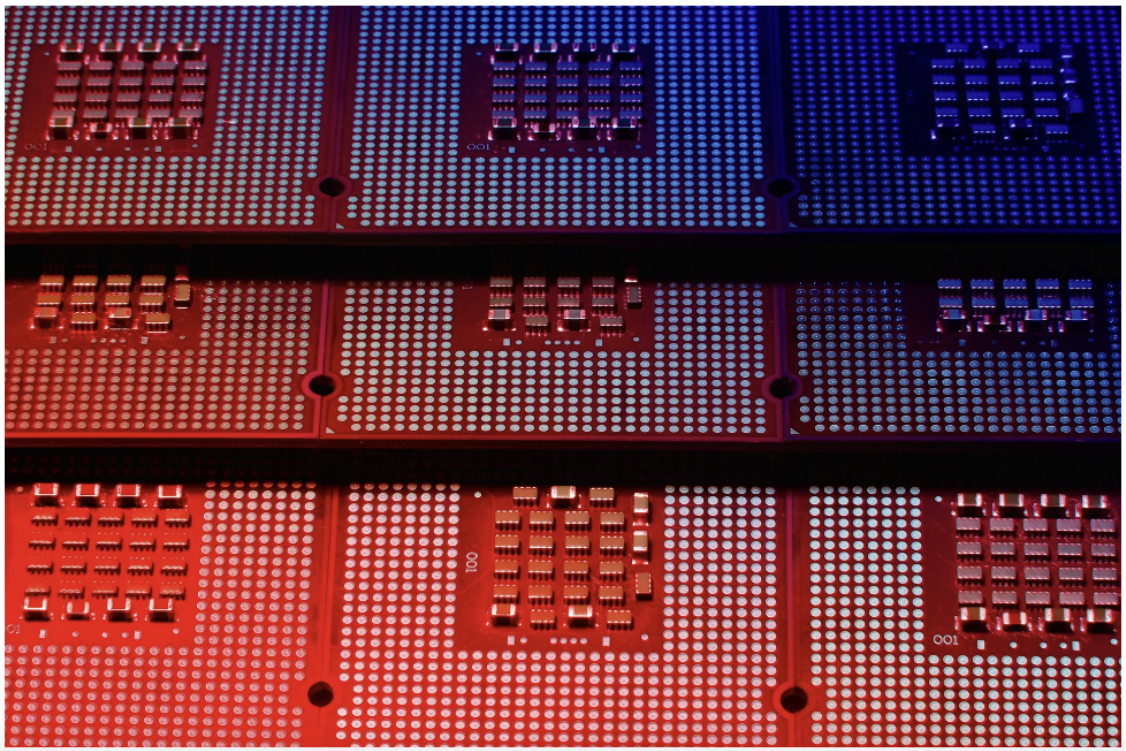
More recently TDX[3]

WIRED BACKCHANNEL BUSINESS CULTURE GEAR IDEAS SCIENCE SECURITY

LILY HAY NEWMAN SECURITY APR 24, 2023 1:12 PM

Intel Let Google Cloud Hack Its New Secure Chips and Found 10 Bugs

To protect its Confidential Computing cloud infrastructure and gain critical insights, Google leans on its relationships with chipmakers.



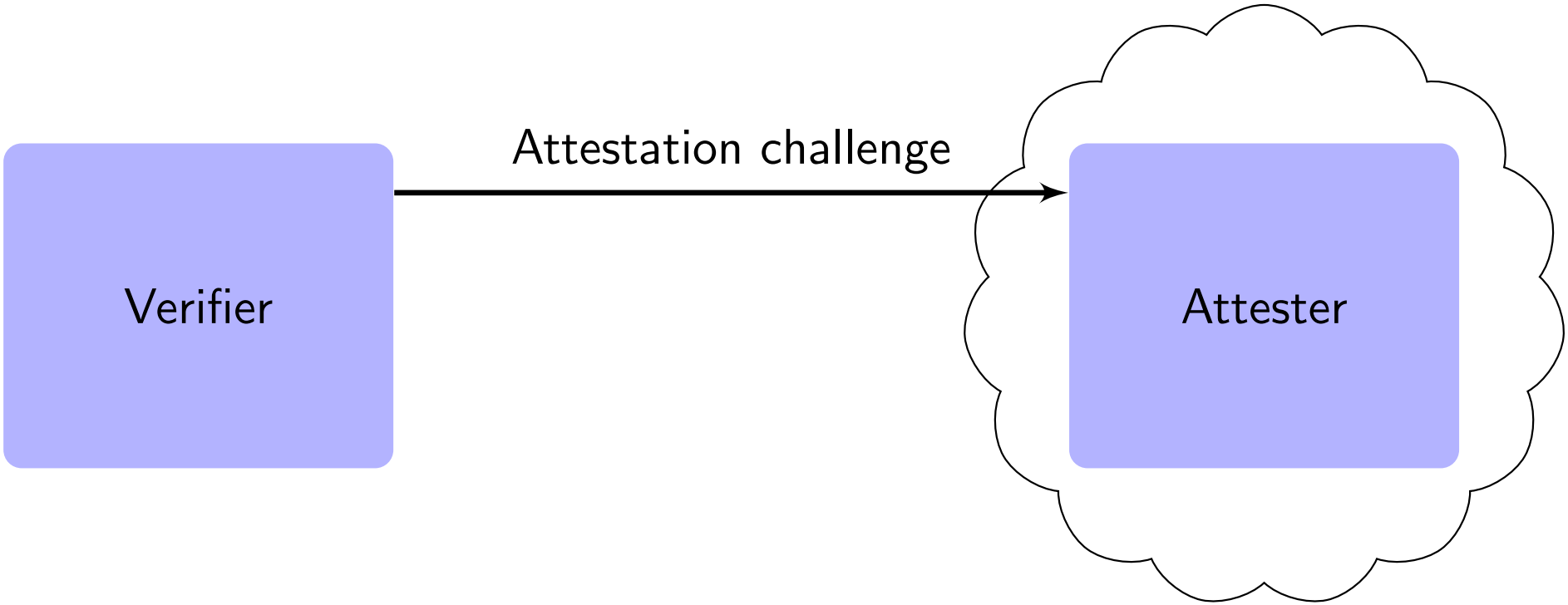
PHOTOGRAPH: GETTY IMAGES

[3]Wired, *Intel Let Google Cloud Hack Its New Secure Chips and Found 10 Bugs*, 2023.

Outline

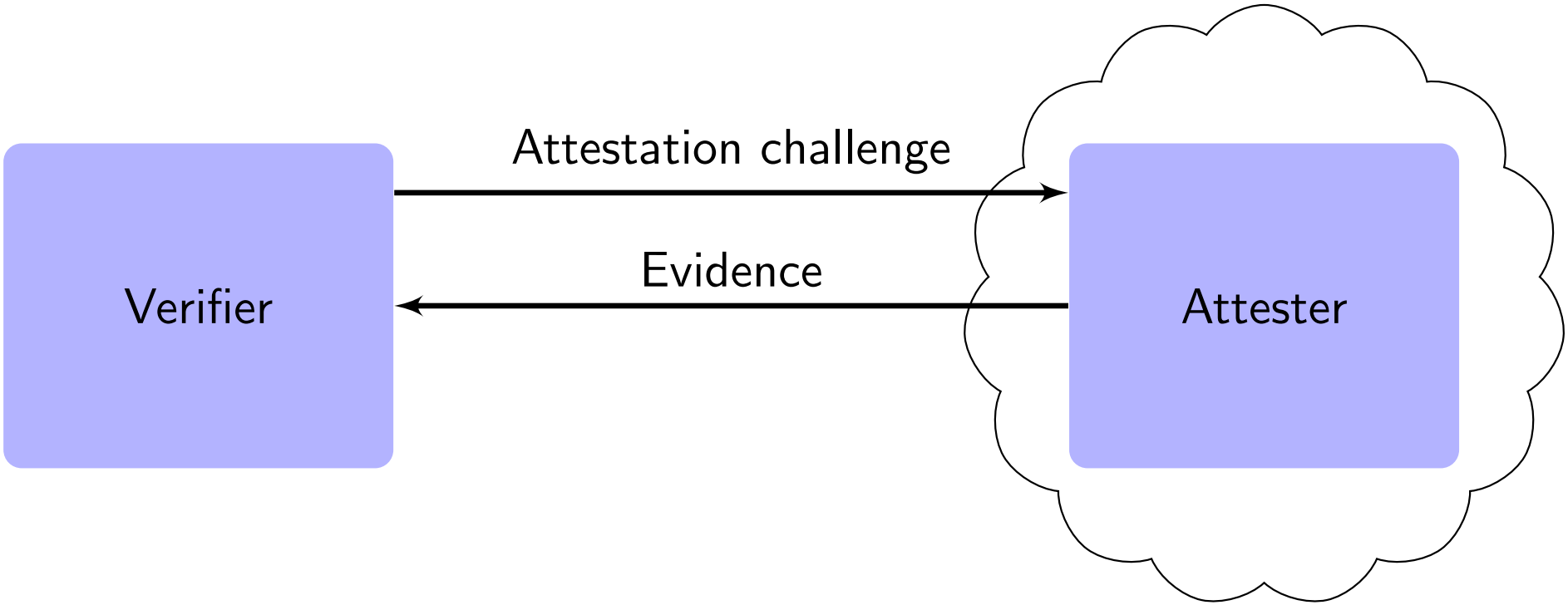
1. Motivation
2. Approach
3. Results
4. Summary

Architecturally-defined Attestation in CC[4]



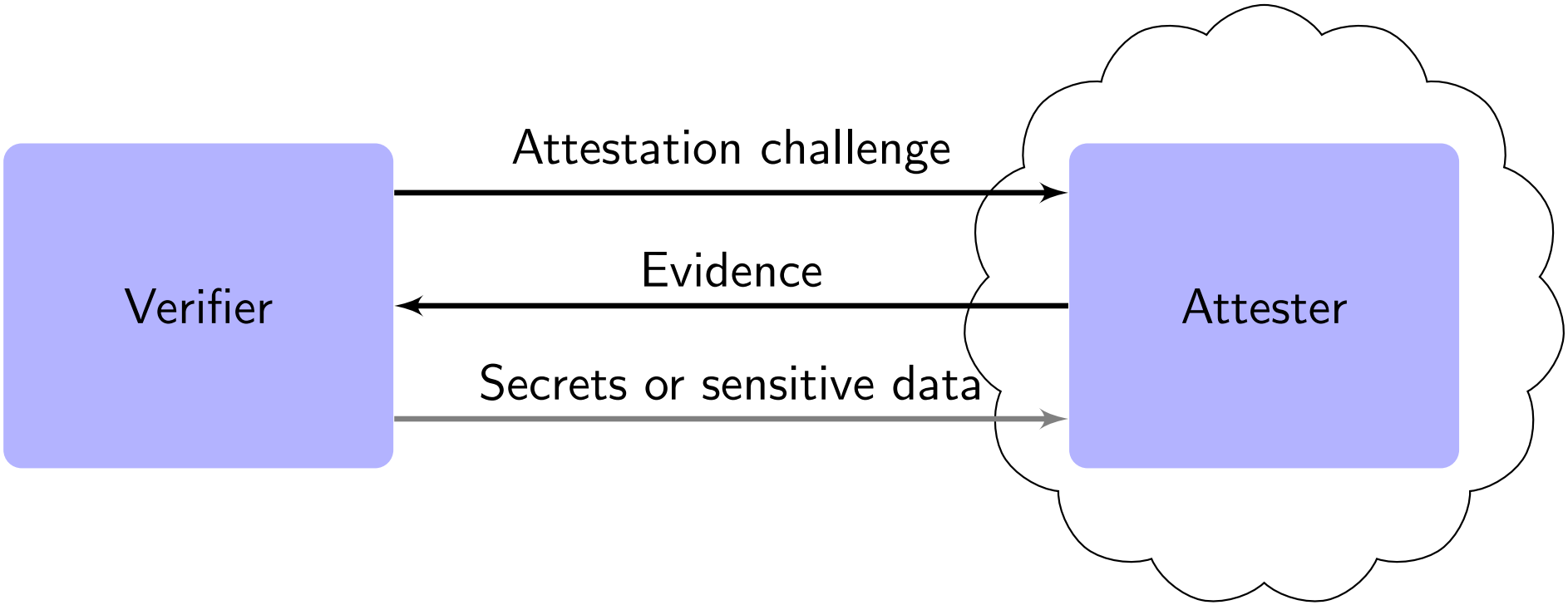
[4] Sardar and Fetzer, *Confidential Computing and Related Technologies : A Review*, 2021.

Architecturally-defined Attestation in CC[4]



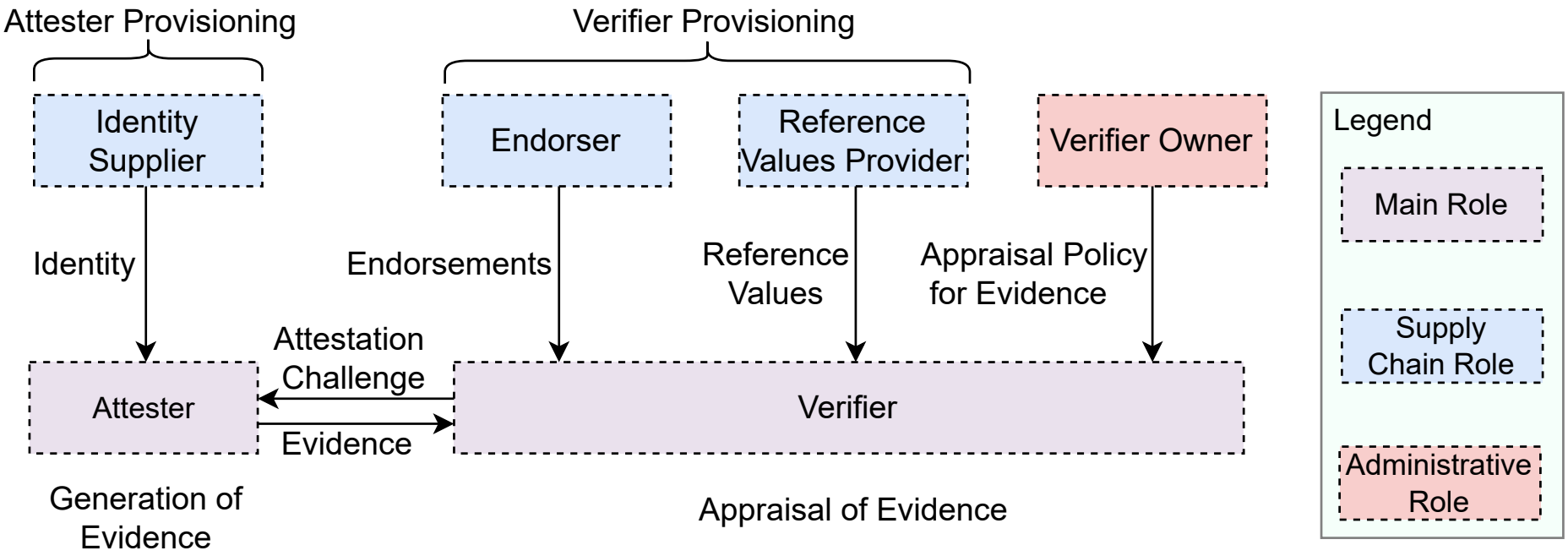
[4] Sardar and Fetzer, *Confidential Computing and Related Technologies : A Review*, 2021.

Architecturally-defined Attestation in CC[4]



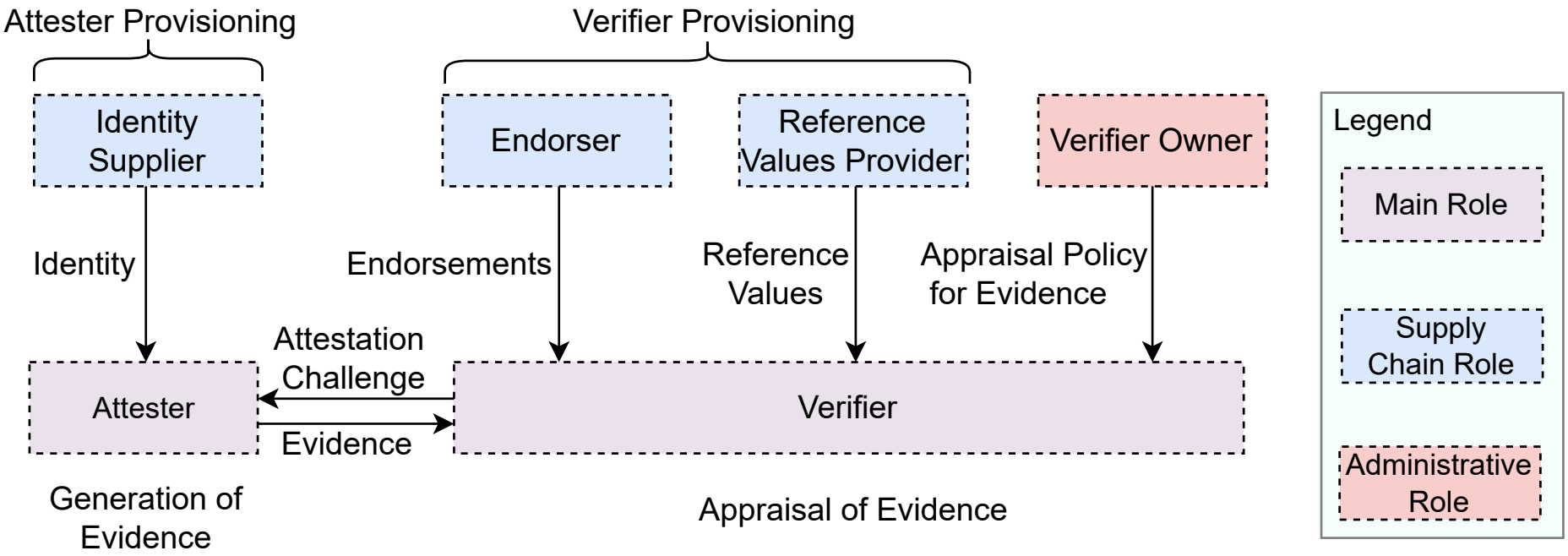
[4] Sardar and Fetzer, *Confidential Computing and Related Technologies : A Review*, 2021.

Architecturally-defined Attestation



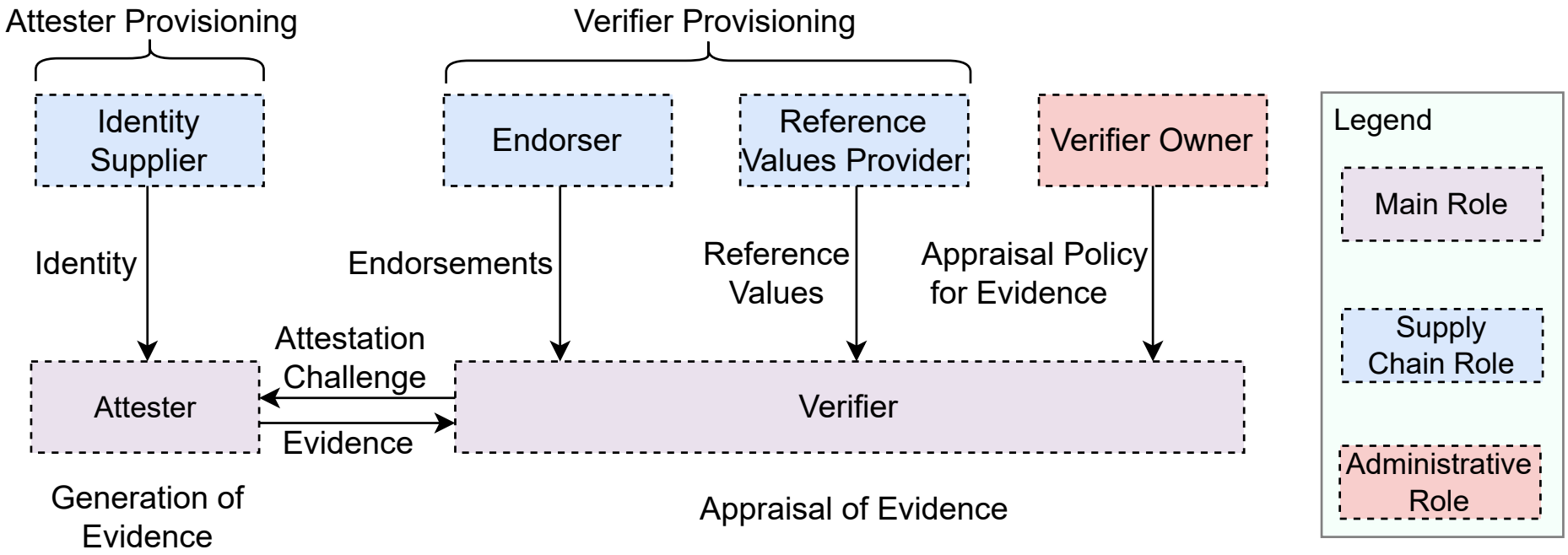
- Holistic coverage of phases

Architecturally-defined Attestation



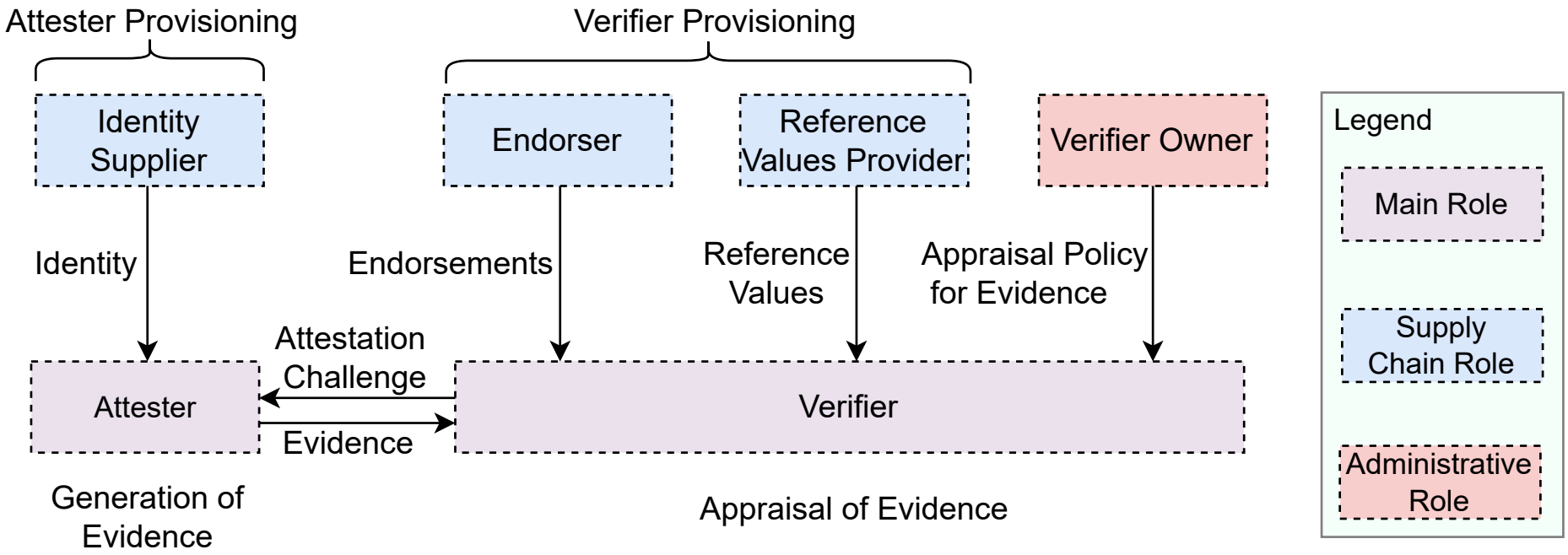
- Holistic coverage of phases
  - Provisioning

Architecturally-defined Attestation



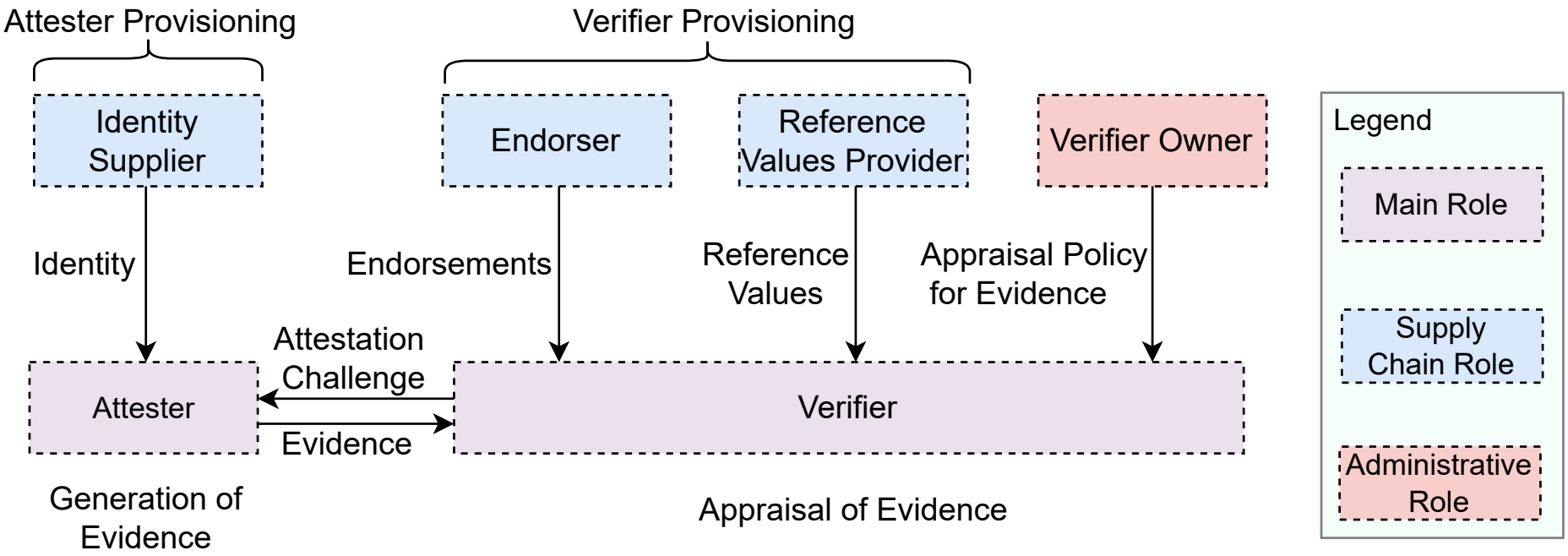
- Holistic coverage of phases
  - Provisioning
    - Attester Provisioning

Architecturally-defined Attestation



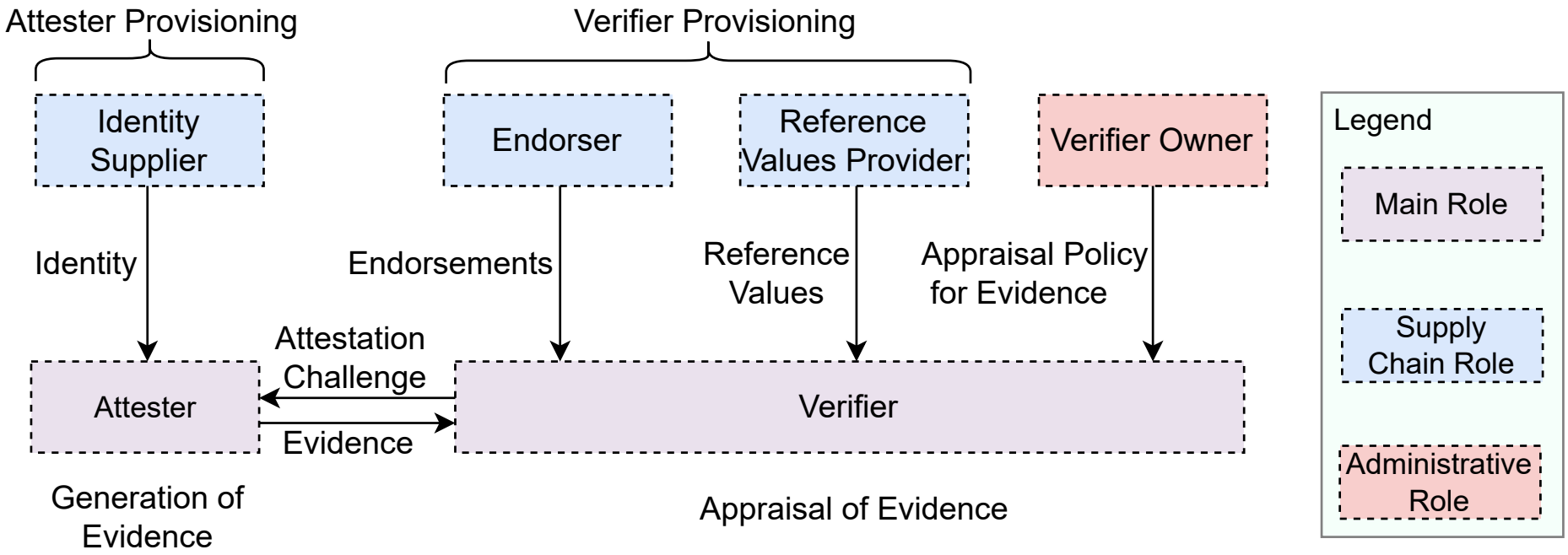
- Holistic coverage of phases
  - Provisioning
    - Attester Provisioning
    - Verifier Provisioning

Architecturally-defined Attestation



- Holistic coverage of phases
  - Provisioning
    - Attester Provisioning
    - Verifier Provisioning
  - Initialization

Architecturally-defined Attestation



- Holistic coverage of phases
  - Provisioning
    - Attester Provisioning
    - Verifier Provisioning
  - Initialization
  - Attestation Protocol

Contributions

- Most detailed formal model of Intel TDX attestation, including

Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain

Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain
  - Verifier steps

Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain
  - Verifier steps
  - Initialization phase

Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain
  - Verifier steps
  - Initialization phase
  - Variable measurements

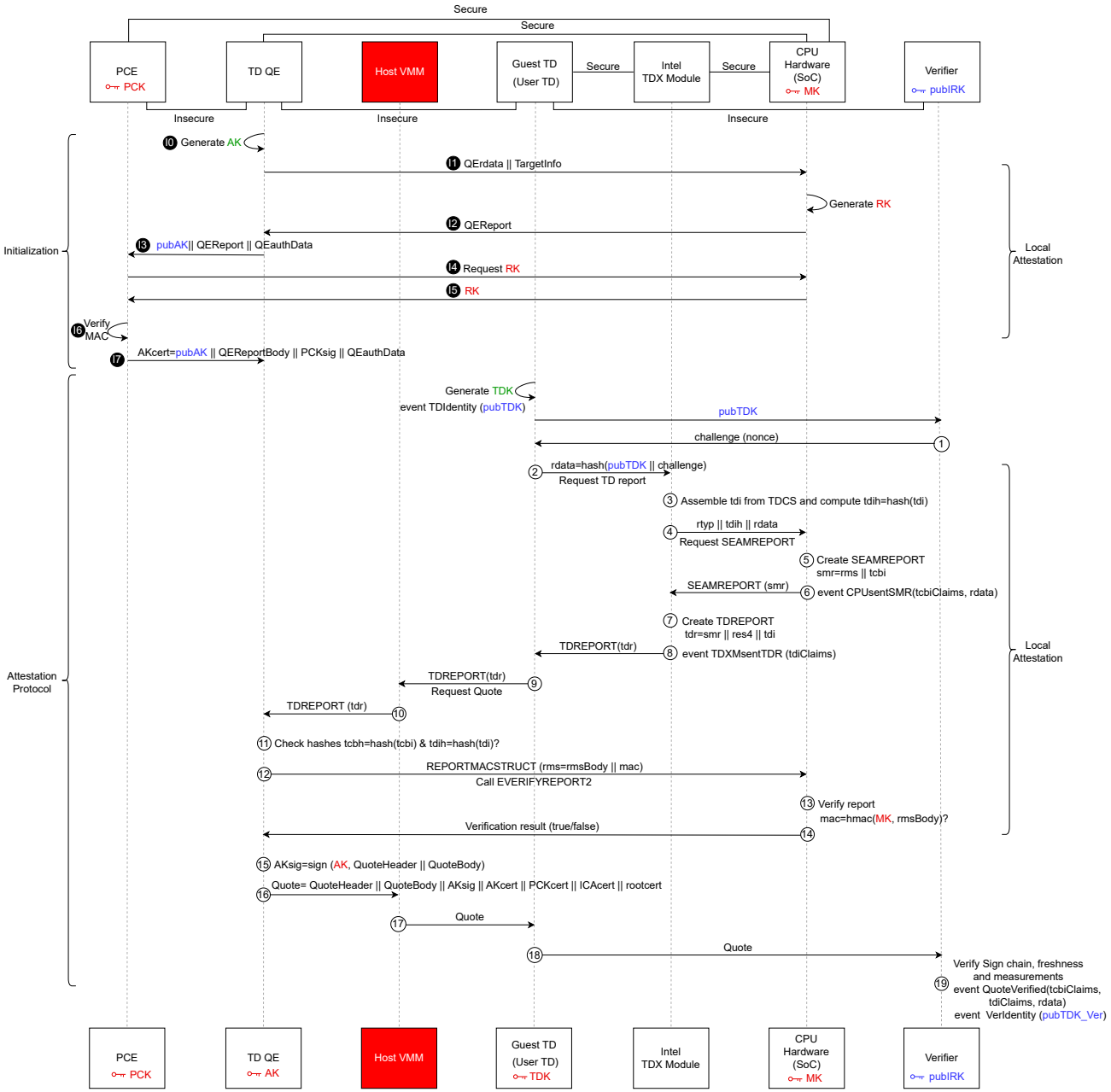
Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain
  - Verifier steps
  - Initialization phase
  - Variable measurements
- Formal proof of insecurity of Intel's claimed TCB

Contributions

- Most detailed formal model of Intel TDX attestation, including
  - Certificate chain
  - Verifier steps
  - Initialization phase
  - Variable measurements
- Formal proof of insecurity of Intel's claimed TCB
- First formal analysis of Arm CCA attestation

TDX Model


Secure
Secure
PCE
PCK
TD QE
Host VMM
Guest TD
(User TD)
Secure
Intel
TDX Module
Secure
CPU
Hardware
(SoC)
MK
Verifier
pubIRK
Insecure
Insecure
Insecure
Initialization
I0 Generate AK
I1 QErdata || TargetInfo
Generate RK
I2 QEReport
I3 pubAK|| QEReport || QEauthData
I4 Request RK
I5 RK
Local
Attestation
I6 Verify
MAC
I7 AKcert=pubAK || QEReportBody || PCKsig || QEauthData
Generate TDK
event TDIdentity (pubTDK)
pubTDK
challenge (nonce)
1
2 rdata=hash(pubTDK || challenge)
Request TD report
3 Assemble tdi from TDCS and compute tdih=hash(tdi)
4 rtyp || tdih || rdata
Request SEAMREPORT
5 Create SEAMREPORT
smr=rms || tcbi
SEAMREPORT (smr)
6 event CPUsentSMR(tcbiClaims, rdata)
7 Create TDREPORT
tdr=smr || res4 || tdi
TDREPORT(tdr)
8 event TDXMsentTDR (tdiClaims)
Local
Attestation
Attestation
Protocol
TDREPORT(tdr)
Request Quote
9
TDREPORT (tdr)
10
11 Check hashes tcbh=hash(tcbi) & tdih=hash(tdi)?
12 REPORTMACSTRUCT (rms=rmsBody || mac)
Call EVERIFYREPORT2
13 Verify report
mac=hmac(MK, rmsBody)?
Verification result (true/false)
14
15 AKsig=sign (AK, QuoteHeader || QuoteBody)
16 Quote= QuoteHeader || QuoteBody || AKsig || AKcert || PCKcert || ICAcert || rootcert
17 Quote
18 Quote
19 Verify Sign chain, freshness
and measurements
event QuoteVerified(tcbiClaims,
tdiClaims, rdata)
event VerIdentity (pubTDK_Ver)
PCE
PCK
TD QE
AK
Host VMM
Guest TD
(User TD)
TDK
Intel
TDX Module
CPU
Hardware
(SoC)
MK
Verifier
pubIRK

Challenge: Complicated designs with vague and outdated specs and very little support[5]

Intel Community / Developer Software Forums / Software Development Technologies /
Intel® Software Guard Extensions (Intel® SGX)

1400 Diskussionen

index 1 in tdxtcbcomponents

Abonnieren | Mehr Aktionen

UsamaS 09-04-2023 • 01:11 AM • 111Aufrufe
Einsteiger

In the "Get TDX TCB Info" flow (https://api.portal.trustedservices.intel.com/documentation#pcs-tcb-info-tdx-v4), step 4 states:

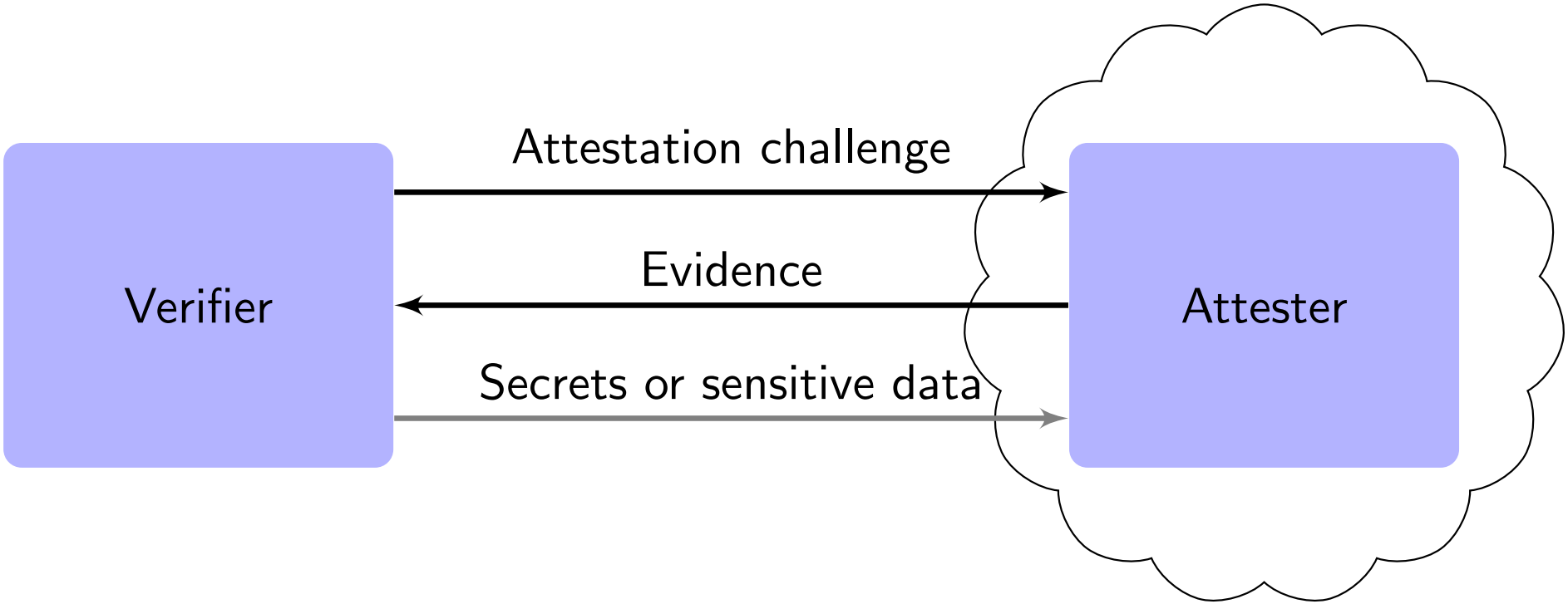
"For the selected TCB level verify that SVN at index 1 in tdxtcbcomponents array matches the value of SVN at index 1 in TEE TCB SVNs array (from TD Report in Quote). In case of a mismatch the selected TCB level should be rejected as TCB Info that was used for the comparison is not supported for this platform configuration."

My question is:

What is so special about index 1 that it requires an equality check? What does index 1 represent? Typically all SVNs have a non-equality check (>=) as in step 3 (a,b,c).

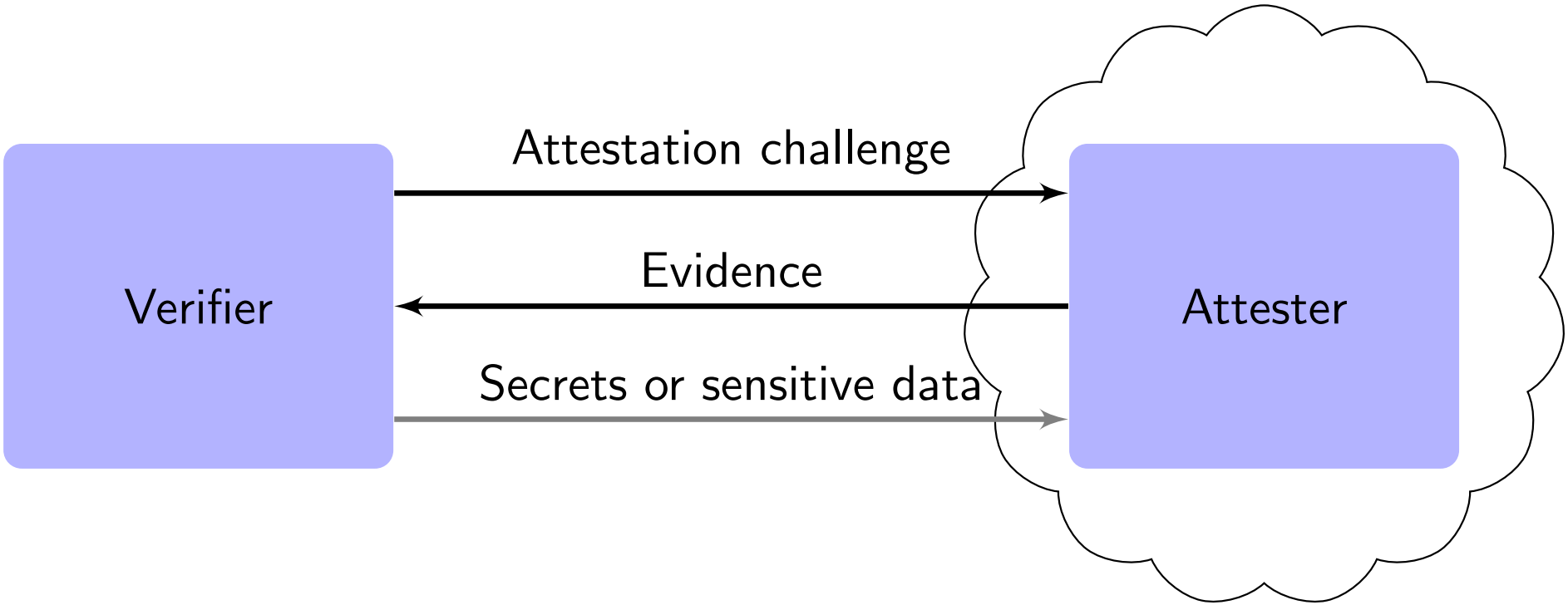
[5] https://community.intel.com/t5/Intel-Software-Guard-Extensions/index-1-in-tdxtcbcomponents/m-p/1520194

Properties



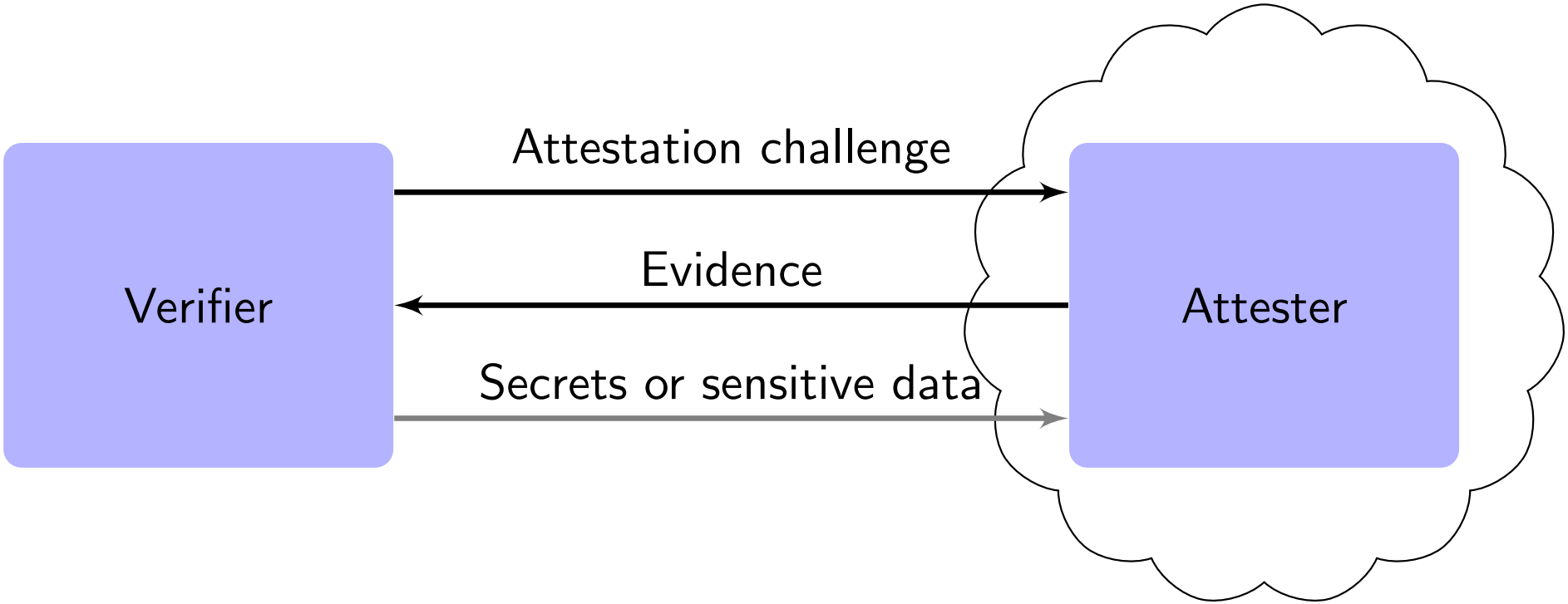
- Sanity checks

Properties



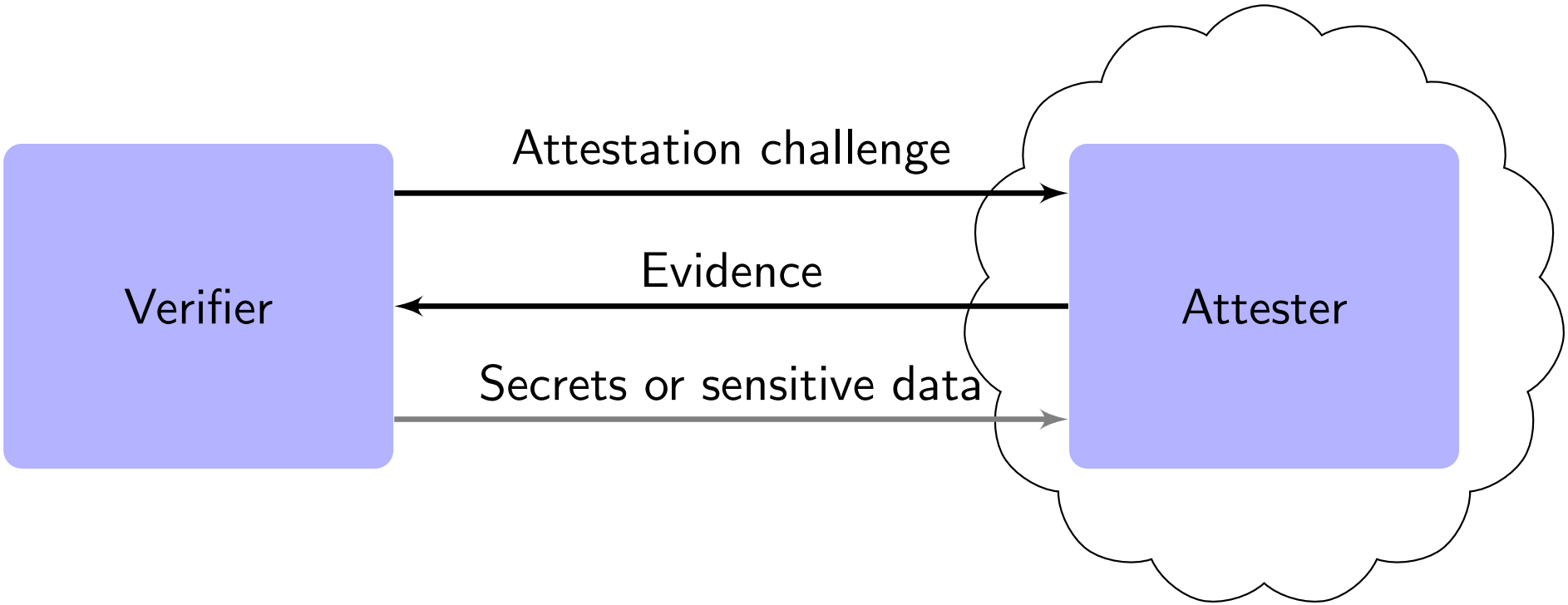
- Sanity checks
- Integrity of Evidence

Properties



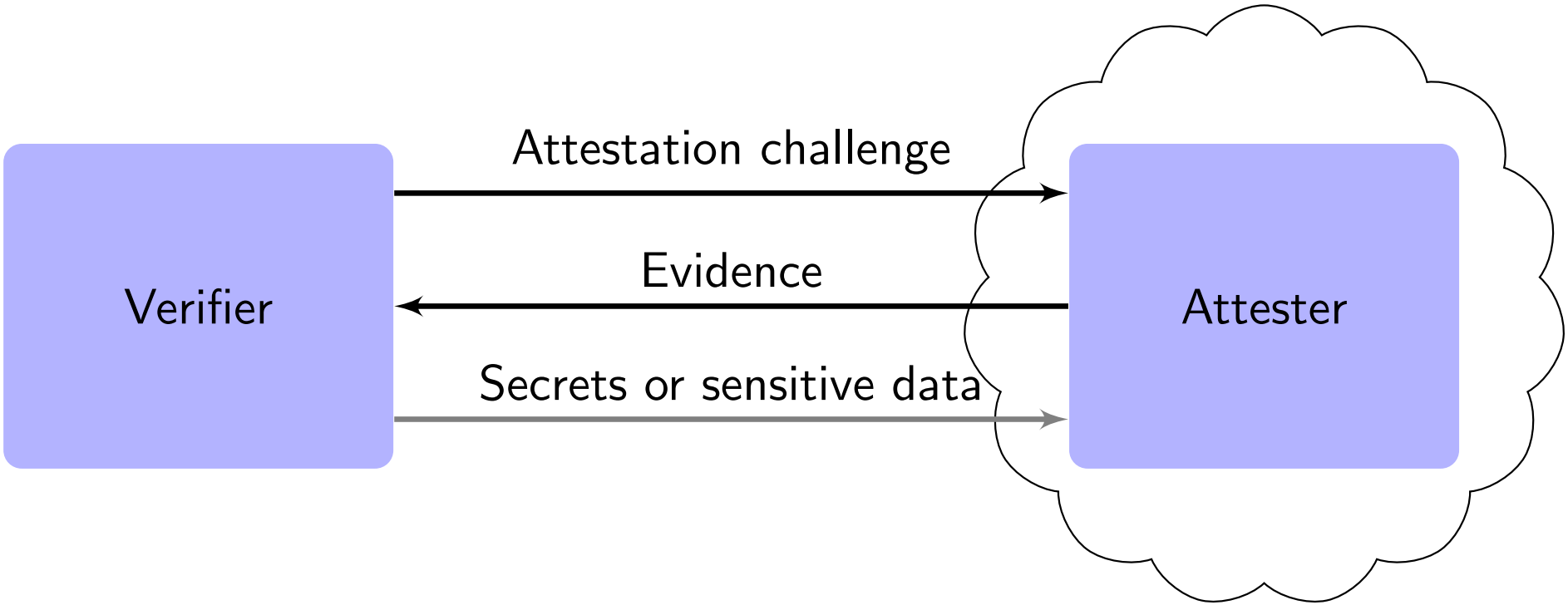
- Sanity checks
- Integrity of Evidence
- Freshness of Evidence

Properties



- Sanity checks
- Integrity of Evidence
- Freshness of Evidence
- Confidentiality/Secrecy of attestation-related keys

Properties



- Sanity checks
- Integrity of Evidence
- Freshness of Evidence
- Confidentiality/Secrecy of attestation-related keys
- Attester Authentication

Outline

1 Motivation

2 Approach

3 Results

4 Summary

TCB Claimed by Intel[6]

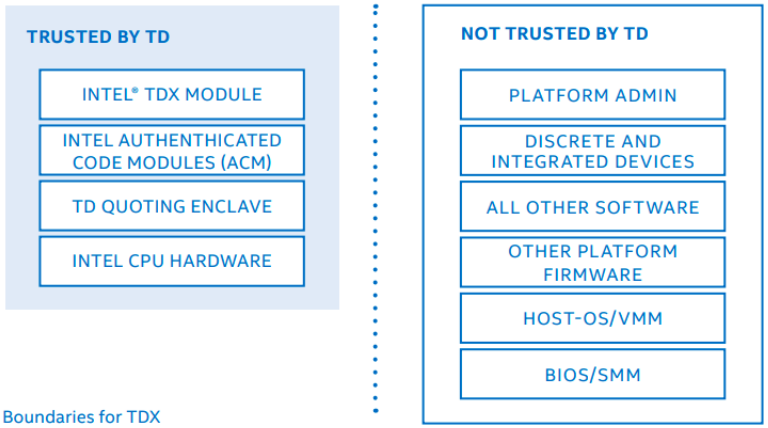
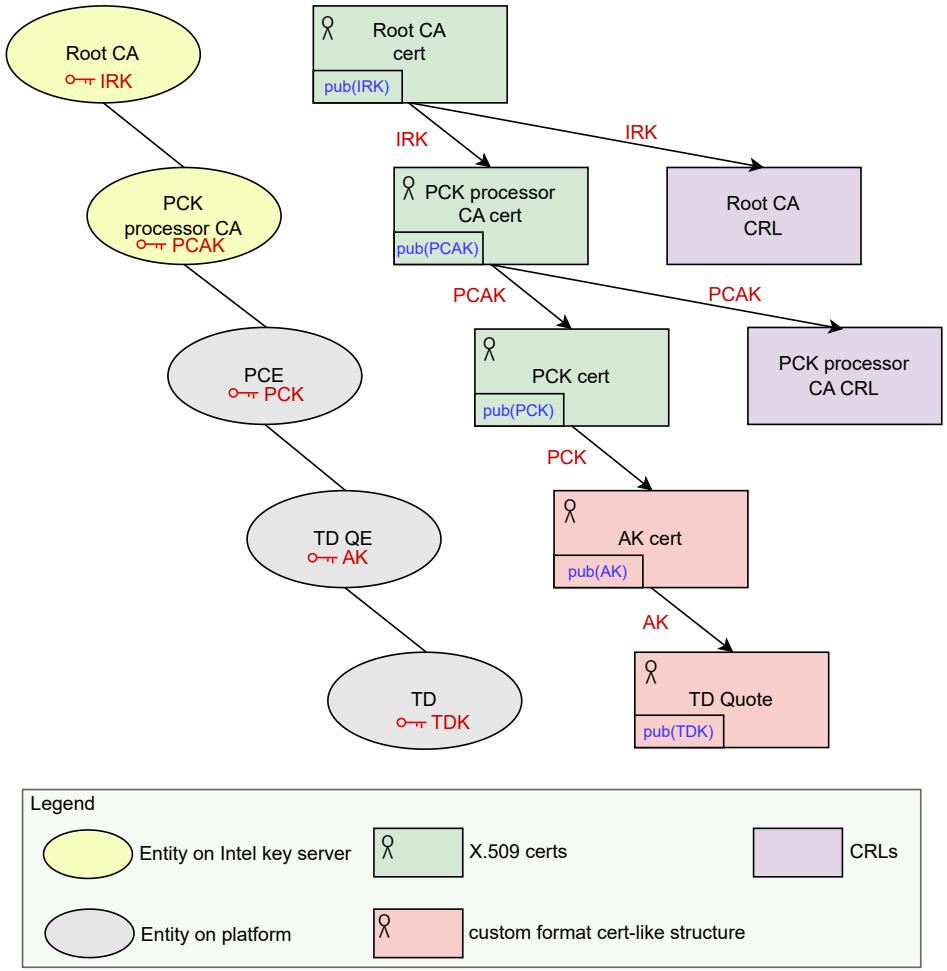


Figure 5.1. Trust Boundaries for TDX



[6] Intel, *Intel ® Trust Domain Extensions*, 2021.

Verification Summary in ProVerif[7]

	Integrity	Freshness	Confidentiality	Authentication
Intel's claimed	×	×	×	×
Our proposed	✓	✓	✓	×

```
------------------------------------------------------------
Verification summary:

Query not event(AKverified(pubAK_1)) is false.

Query not event(CPUsentSMR(tcbiClaims_1,rdata_1)) is false.

Query not event(TDXMsentTDR(tdiClaims_1)) is false.

Query not event(QuoteVerified(tcbiClaims_1,tdiClaims_1,rdata_1)) is false.

Query not (event(TDidentity(pubTDK_1)) && event(VerIdentity(pubTDK_Ver_1))) is false.

Query event(AKverified(pubAK_1)) ==> event(AKsent(pubAK_1)) is true.

Query event(QuoteVerified(tcbiClaims_1,tdiClaims_1,rdata_1)) ==> event(CPUsentSMR(tcbiClaims_1,rdata_1)) is false.

Query event(QuoteVerified(tcbiClaims_1,tdiClaims_1,rdata_1)) ==> event(TDXMsentTDR(tdiClaims_1)) is false.

Query inj-event(QuoteVerified(tcbiClaims_1,tdiClaims_1,rdata_1)) ==> inj-event(CPUsentSMR(tcbiClaims_1,rdata_1)) is false.

Query inj-event(QuoteVerified(tcbiClaims_1,tdiClaims_1,rdata_1)) ==> inj-event(TDXMsentTDR(tdiClaims_1)) is false.

Query secret PCK_1,PCK is false.

Query secret PCAK is true.

Query secret AK_2,AK_1,AK is true.

Query secret MK_1,MK is true.

Query event(AKverified(pubAK_PCE_1)) && event(AKsent(pubAK_1)) ==> pubAK_PCE_1 = pubAK_1 is true.

Query event(VerIdentity(pubTDK_Ver_1)) && event(TDidentity(pubTDK_1)) ==> pubTDK_1 = pubTDK_Ver_1 is false.

------------------------------------------------------------


real    0m55,648s
user    0m55,432s
sys     0m0,132s
```

[7] Blanchet, Cheval, and Cortier, "ProVerif with lemmas, induction, fast subsumption, and much more", 2022.

Outline

1 Motivation

2 Approach

3 Results

4 Summary

Take-home

- Identified grey areas in RATS (e.g., Endorsements)

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions
  - How to verify the Verifier?

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions
  - How to verify the Verifier?
  - How to verify the runtime configurations?

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions
  - How to verify the Verifier?
  - How to verify the runtime configurations?
  - Is RATS architecture better than split architecture?

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions
  - How to verify the Verifier?
  - How to verify the runtime configurations?
  - Is RATS architecture better than split architecture?

Take-home

- Identified grey areas in RATS (e.g., Endorsements)
  - can be more precise! Thanks to Dave for pursuing this!
- Need for systematic design of attestation protocols
- Open questions
  - How to verify the Verifier?
  - How to verify the runtime configurations?
  - Is RATS architecture better than split architecture?

Trusted until formally verified!

Next steps

- TEEP WG

[8] https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1

Next steps

- TEEP WG
  - Found a problem[8] in FV of TEEP during hackathon

[8] https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1

Next steps

- TEEP WG
  - Found a problem[8] in FV of TEEP during hackathon
  - Integrate RA with TEEP

[8] https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1

Next steps

- TEEP WG
  - Found a problem[8] in FV of TEEP during hackathon
  - Integrate RA with TEEP
- UFMRG: Sample problem

[8] https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1

Next steps

- TEEP WG
  - Found a problem[8] in FV of TEEP during hackathon
  - Integrate RA with TEEP
- UFMRG: Sample problem
- TLS WG

[8] https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1

Next steps

- TEEP WG
  - Found a problem[8] in FV of TEEP during hackathon
  - Integrate RA with TEEP
- UFMRG: Sample problem
- TLS WG
  - Attested TLS
    https://datatracker.ietf.org/doc/draft-fossati-tls-attestation/

[8] https://github.com/tetsuya-okuda-hco/public-teep-formal-verif/issues/1

Key References

Birkholz, Henk et al. *Remote ATtestation procedureS (RATS) Architecture*. RFC 9334. Jan. 2023. DOI: 10.17487/RFC9334. URL: https://www.rfc-editor.org/info/rfc9334.

Blanchet, Bruno, Vincent Cheval, and Véronique Cortier. "ProVerif with lemmas, induction, fast subsumption, and much more". In: *IEEE Symposium on Security and Privacy (S&P'22)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2022, pp. 205–222. DOI: 10.1109/SP46214.2022.00013.

Intel. *Intel ® Trust Domain Extensions*. Aug. 2021. URL: https://cdrdv2.intel.com/v1/dl/getContent/690419.

Sardar, Muhammad Usama and Christof Fetzer. *Confidential Computing and Related Technologies : A Review*. 2021. URL: https://www.researchgate.net/publication/356474602_Confidential_Computing_and_Related_Technologies_A_Review.

Wired. *Intel Let Google Cloud Hack Its New Secure Chips and Found 10 Bugs*. 2023. URL: https://www.wired.com/story/intel-google-cloud-chip-security/ (visited on 04/25/2023).

Call to Action

- Bring your expertise:
  https://github.com/CCC-Attestation/formal-spec-TEE
- Additional information: link here

- Paper on formal verification coming soon