Secure Evidence and Attestation Transport
Internet-Draft
Intended status: Informational
Expires: 24 July 2026

M. U. Sardar
TU Dresden

20 January 2026

Pre-, Intra- and Post-handshake Attestation
draft-usama-seat-intra-vs-post-02

Abstract

This document presents a taxonomy of extending TLS protocol with remote attestation, referred to as attested TLS. It also presents high-level analysis of benefits and limitations of each category, namely pre-handshake attestation, intra-handshake attestation and post-handshake attestation.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://muhammad-usama-sardar.github.io/seat-intra-vs-post/draft-usama-seat-intra-vs-post.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-usama-seat-intra-vs-post/.

Discussion of this document takes place on the Secure Evidence and Attestation Transport Working Group mailing list (mailto:seat@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/seat. Subscribe at https://www.ietf.org/mailman/listinfo/seat/.

Source for this draft and an issue tracker can be found at https://github.com/muhammad-usama-sardar/seat-intra-vs-post.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

Based on our extensive analysis of attested TLS [Tech-Concepts], we classify attested TLS into three main categories:

* pre-handshake attestation,

* intra-handshake attestation, and

* post-handshake attestation.

In pre-handshake attestation, the signing of Claims [Tech-Concepts] precedes the TLS handshake, while post-handshake attestation applies the reverse. Intra-handshake attestation requires the signing of Claims to be done within the TLS handshake protocol.

1.1. Scope

In this version, we analyze the three categories (without combinations) with a focus on the last two, i.e., intra-handshake attestation and post-handshake attestation.

The current scope of this draft is existing specifications and real-world implementations pointed in the given references. Any theoretical solutions are currently out of scope until some specification or implementation emerges.

For simplicity, we consider simple Attester with only one Attesting Environment and only one Target Environment [RFC9334]. That is, complicated scenarios such as Composite Device [RFC9334] etc. are out of scope in this version.

From RATS perspective, we consider Background Check Model [RFC9334]. Future versions will add Passport Model [RFC9334].

From TLS perspective, the scope is limited to TLS 1.3 as per [SEAT-Charter]. That is, older versions of TLS are explicitly out of scope.

1.2. Note

Regarding remote attestation, we note that:

> Remote attestation provides guarantees about the state of Attester *only* at the time at which signing of Claims is done to generate Evidence [Tech-Concepts].

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

We use terminology from [RFC9334] and [I-D.ietf-tls-rfc8446bis] slightly loosely (intentionally) for readability. Future versions will tighten it.

In addition, we define three temporal terms:

* *Evidence Generation Time*: Time when Evidence is generated (more specifically when Claims are signed)

* *Connection Establishment Time*: Time at which TLS handshake is performed

* *Lifetime of Connection*: Time period starting from Connection Establishment Time until the connection exists.

3. Pre-handshake Attestation

Since the Evidence Generation Time could be at any arbitrary point of time in the past compared to the Connection Establishment Time, pre-handshake attestation provides no guarantees about the state of Attester at the Connection Establishment Time and during the Lifetime of Connection.

4. Intra-handshake Attestation

Intra-handshake attestation improves the situation where Evidence Generation Time is the same as Connection Establishment Time.

In following subsections, we present the benefits and limitations of intra-handshake attestation.

4.1. Benefits

4.1.1. No Additional Application-Level Protocol

Intra-handshake attestation does not require a new application-layer protocol or message exchange. Evidence and related metadata are conveyed within handshake via TLS extensions. TLS is responsible for conveyance of the Evidence; it does not perform appraisal of Evidence or authorization. Appraisal of Evidence, policy evaluation, and trust decisions are performed by application-level components that consume the attestation properties exposed by the TLS stack. As a result, while no new application-layer protocol is required, applications do incorporate additional trust logic to interpret attested connection properties and make security-relevant decisions.

Related to this, Markus Rudy shares his practical experience [Markus-16Jan]:

> Conveying the evidence is not enough, it needs to be verified as well in order to end up with a trustworthy channel. We decided to integrate verification into the handshake, too, but that has massive drawbacks: Verification can take orders of magnitude longer than normal TLS handshakes, and usually involves remote calls, affecting all sorts of timeouts. However, doing the verification at the application level would require forwarding information from the handshake (e.g. nonce), at which point the application needs to be fully aware of the handshake protocol in order to verify it, breaking the intended layering.

4.1.2. Avoid Extra Round Trips for One-time Attestation

It is claimed that intra-handshake attestation avoids extra round trips for use cases which require remote attestation only once during Connection Establishment Time.

However, this may only be valid in cases when the Connection Establishment Time without remote attestation is significantly higher than the time for generation and appraisal of Evidence. For instance, Markus Rudy shares his practical experience [Markus-16Jan]:

> I don’t think saving extra roundtrips is an appropriate design goal when attestation is required. Generating evidence alone takes much longer than normal network roundtrip times, not even speaking of verification.

On request, he kindly conducted an experiment and shared his preliminary results of experiment based on attested TLS implementation in Edgeless Systems Contrast where Coordinator is one of the components [Markus-19Jan]:

> I did a quick experiment in our testing lab, running on the same machine as the Coordinator:
>
> - TCP connections are local, and thus the TCP connection establishment unsurprisingly takes only 0.5ms. But even to neighbouring nodes in the same cluster, the TCP handshake takes below 2ms.
>
> - I measured generation of evidence including the TLS session establishment, but with these numbers I don’t think it makes a lot of difference:
>
> o SNP: Median time of 140ms from TCP SYN to TLS channel established and evidence sent to the client.
>
> o TDX: Median time of 1020ms, same procedure. I don’t know why it is that slow, it should only be making machine-local remote calls, if any.
>
> - So far, I only managed to measure TDX verification, which adds another 340ms. This is bound by remote HTTP requests, afaiu, and could be optimized with locally cached collateral, CRL, etc. I’d expect SNP to exhibit similar timing, because verification does similar remote calls.

We summarize that in the following table:

Property	Intel TDX	AMD SEV-SNP
Generation of Evidence + TLS	1020	140
Appraisal of Evidence	340	not available (expected ca. 340)

Table 1: Preliminary analysis by Markus Rudy (Median time in ms)

4.2. Limitations

4.2.1. Limited Claims Availability

Since limited Claims are available at the Evidence Generation Time, it does not provide complete security posture of the Attester, such as runtime integrity of Attester.

4.2.2. Invasive Changes in TLS

To be made secure, it requires invasive changes in TLS protocol, as deep as key schedule and adding or modifying existing handshake messages [ID-Crisis], which are explicitly out of scope of [SEAT-Charter]:

> The attested (D)TLS protocol extension will not modify the (D)TLS protocol itself. It may define (D)TLS extensions to support its goals but will not modify, add, or remove any existing protocol messages or modify the key schedule.

4.2.3. State After Connection Establishment Not Covered

It provides no guarantees about the state of Attester during the lifetime of connection. This is a security concern in long-lived connections where state of Attester may change after Connection Establishment Time. Note that session resumption is a new connection [I-D.ietf-tls-rfc8446bis].

4.2.4. High Handshake Latency

Because of signature in Evidence generation and verification of signatures during appraisal, this leads to high handshake latency. This may not be desirable for some applications.

Markus Rudy shares his practical experience [Markus-16Jan]:

> | Conveying the evidence is not enough, it needs to be verified as
> | well in order to end up with a trustworthy channel. We decided to
> | integrate verification into the handshake, too, but that has
> | massive drawbacks: Verification can take orders of magnitude
> | longer than normal TLS handshakes, and usually involves remote
> | calls, affecting all sorts of timeouts. However, doing the
> | verification at the application level would require forwarding
> | information from the handshake (e.g. nonce), at which point the
> | application needs to be fully aware of the handshake protocol in
> | order to verify it, breaking the intended layering.

4.2.5. Maturity of TEEs

With several attacks (see Section 8), attestation in TEEs may not yet be mature enough to be integrated _within_ TLS handshake.

Ayoub Benaissa remarks [Ayoub-16Jan]:

> | TLS might not be well suited to include this in its protocol. Not
> | sure TEEs are even as mature for the people to see that it should
> | be included right now. The plan to make it a post-handshake
> | protocol makes more sense right now. A future where it’s
> | incorporated into TLS might exist, but I don’t think there is
> | enough motivation right now.

4.2.6. Amount of Effort

Markus Rudy shares his practical experience [Markus-16Jan]:

> | "Keeping attestation out of the application logic" is not as
> | straightforward as it sounds. In the background-check model, the
> | attester needs to collect evidence in response to the relying
> | party’s challenge (nonce). We were lucky that the Golang TLS
> | stack can be supplied with arbitrary closures that are called
> | during the handshake, but in my experience this is a rare design
> | choice and may also be difficult to implement in other languages.

Ayoub Benaissa remarks [Ayoub-16Jan]:

> | An intra-handshake requires much more work compared to a post-
> | handshake. People need to agree on how to add this as optional in
> | TLS (we can’t force everyone to use it of course), the standard
> | needs to be implemented by major libraries, and then it will be
> | available in major client/server applications. If any of the
> | prior steps doesn’t go through, it means you have to patch your
> | components to make it work, which is not convenient / less secure.

4.2.7. Difficulty of Debugging Attestation

Markus Rudy shares his practical experience [Markus-16Jan]:

> There’s only so much information in a TLS alert message, and it’s definitely not enough to understand remote verification failures. While I understand this to be a deliberate design choice by TLS, I found this to be a hindrance for operating and debugging a large number of services in practice.

5. Post-handshake Attestation

Post-handshake attestation improves the situation further by signing the Claims during Lifetime of Connection, i.e., at the time when it is actually required. Hence, together with use cases requiring one-time attestation, it covers the use cases of long-lived connections requiring re-attestation. For post-handshake attestation, first round of remote attestation MUST be done immediately after Connection Establishment Time, and Relying Party (RP) [RFC9334] MUST not send any secure data until Evidence is successfully appraised.

In following subsections, we present the benefits and limitations of post-handshake attestation.

5.1. Benefits

In general, it allows re-authentication and re-attestation without tearing down the connection.

5.1.1. Full Claims Availability

Since all Claims are available at the time of post-handshake attestation (during Lifetime of Connection), it provides complete security posture of the Attester.

5.1.2. No Change in TLS

It does not require any change in TLS protocol.

5.1.3. State After Connection Establishment Is Covered

It provides guarantees about the state of Attester during the Lifetime of Connection. This is particularly helpful in long-lived connections where state of Attester may change after Connection Establishment Time.

5.1.4. Standard Handshake Latency

Since the signature in Evidence generation and verification of signatures during appraisal happen after Connection Establishment Time, there is no additional latency.

5.1.5. Avoid Extra Round Trips

Except for first round of remote attestation, post-handshake attestation outperforms the intra-handshake attestation (one round trip), which requires re-establishing the connection (1.5 round trip).

5.1.6. Ease of Implementation

Ayoub Benaissa remarks [Ayoub-16Jan]:

> We already implemented a post-handshake protocol and have a full demo working. We were able to do this in a matter of weeks. That’s because you don’t need to modify any TLS implementation, but only add a few verification steps after the usual TLS handshake. This is almost the same on the client and server side.

5.1.7. Ease of Verification and Audit

Post-handshake attestation has relatively easier formal analysis and verification. The same may apply to audit.

Markus Rudy remarks [Markus-16Jan]:

> (Formal) verification of a protocol and audit of its implementations might be much easier if it ran on top of TLS. Existing proofs and certifications would not need to be reevaluated.

5.1.8. General Solution for Other Protocols

In post-handshake attestation, design, verification and audit effort will be one-time and any protocol (e.g., Noise) which has support for exporters can then use it without changing each and every protocol.

Markus Rudy shares this requirement [Markus-16Jan]:

> It should be possible to port the general shape of a post-handshake attested TLS protocol to other protocols that provide secure channels and session binding (Noise comes to mind).

5.2. Limitations

5.2.1. Impact on Application Layer

Post-handshake attestation may require changes at the application layer. However, changes at the application layer do not necessarily imply modifications to application business logic or data exchange protocols. Attestation-related functionality may be realized via application-level signalling (Exported Authenticators [RFC9261]) and trust logic, which may be implemented in intermediary components (e.g., proxies, sidecars, or middleware) on both client and server sides. These components are responsible for exchanging and appraising attestation evidence and enforcing trust or authorization decisions before application data is processed. This is analogous to common production deployments in which TLS termination and certificate handling are performed by a fronting proxy, while the application itself remains unchanged and resides behind it.

6. Need for Post-handshake Attestation

We argue that post-handshake attestation is unavoidable (e.g., re-attestation to track changes after Connection Establishment Time for long-lived connections). Use cases where pre-handshake attestation and intra-handshake attestation are insufficient include AI agents/ agentic AI [I-D.jiang-seat-dynamic-attestation].

Intra-handshake attestation only adds unnecessary complexity which is avoidable. All identified use cases [I-D.mihalcea-seat-use-cases] where intra-handshake attestation seems suitable can be covered by post-handshake attestation (by doing attestation round immediately after Connection Establishment Time) but not the other way around.

6.1. IoT Constraints

[SEAT-Charter] includes TLS client as RATS Attester. Client could be a low-power IoT device. There are use cases where periodic or on-demand attestation is required, such as periodic attestation for long-lived, low-power IoT devices or in IoT swarms that need to synchronize software versions before coordinated operations or after configuration updates.

Moreover, we note some observations from LAKE WG:

Michael Richardson shares his insight [MCR-LAKE]:

> I have a half-written document on putting EAT into the full BRSKI protocol. A reason that I stopped is that I realized that doing security posture evaluation at onboarding time (only) wasn’t

> | enough. It has to be done regularly. So having a protocol used
> | at onboarding time and another one during normal operation meant
> | that the onboarding one would have bugs that never get fixed,
> | since the code only runs once.

He further shares [MCR-LAKE2]:

> | My contention, which I think the group agreed with, is that one
> | probably wants to do continuous assurance, that is, to repeat the
> | remote attestation.
>
> Do you want to have two protocols and two code paths? (redundant code in a constrained device?). I suggested that _maybe_ the remote attestation should use it's own /.well-known Path, and that it would just occur after onboarding, and regularly onwards. Maybe it's weird to onboard a device only to kick it out again immediately because it failed remote attestation, but given continuous assurance, this could happen at any time.

Göran Selander observes [Goran-LAKE]:

> | Indeed, if the authentication procedure is repeated at a later
> | stage, for whatever reason, e.g. key rotation, it should be
> | possible to repeat the attestation procedure.

7. Existing Implementations

7.1. Intra-handshake Attestation

Prominent implementations of intra-handshake attestation are all vulnerable to relay attacks [RelayAttacks]. Some of them are abusing the extensions of TLS, such as SNI and ALPN, for conveyance of attestation nonce [RelayAttacks].

7.2. Post-handshake Attestation

Google [Keith-STET-CCC], Microsoft [Stunes-vTPM-CCC], and SCONE [SoK-Attestation] are all using post-handshake attestation.

8. Security Considerations

Most of the document is about security considerations. Also, Security Considerations of [RFC9334] and [I-D.ietf-tls-rfc8446bis] apply. In addition:

* Pre-handshake attestation is vulnerable to *replay* [RA-TLS] and *diversion* [ID-Crisis] attacks. Moreover, pre-handshake attestation leads to a single point of failure.

* Without significant changes to the TLS protocol: Intra-handshake attestation is vulnerable to *diversion* attacks [ID-Crisis]. We reported these attacks to TLS WG in February 2025 [Usama-TLS-26Feb25]. A formal proof is available [ID-Crisis-Repo] for further research and development. Since reporting to TLS WG, these attacks have been practically exploited in TEE.fail (https://tee.fail/), Wiretap.fail (https://wiretap.fail/), and BadRAM (https://badram.eu/). More recently, we found that intra-handshake attestation also does not bind the Evidence to the application traffic secrets, resulting in *relay* attacks [RelayAttacks].

* No attacks on post-handshake attestation are currently known. Post-handshake attestation avoids replay attacks by using fresh attestation nonce. Moreover, it avoids diversion and relay attacks by binding the Evidence to the underlying TLS connection, such as using Exported Keying Material (EKM) [I-D.ietf-tls-rfc8446bis], as proposed in Section 9.2 of [ID-Crisis]. [RFC9261] and [RFC9266] provide mechanisms for such bindings. Efforts for a formal proof of security of post-handshake attestation are ongoing.

8.1. Exploit of Sensitive Hardware-level Information

From the view of the TLS server, post-handshake attestation offers better security than intra-handshake attestation when the server acts as the Attester. In intra-handshake attestation, due to the inherent asymmetry of the TLS protocol, a malicious TLS client could potentially retrieve sensitive hardware-level information from the Evidence *without the client’s trustworthiness (i.e., authentication) first being established by the server*. This information (e.g., vulnerable firmware version) can be exploited for attacks. In post-handshake attestation, the server can ask for client authentication and only send the Evidence after successful client authentication.

9. IANA Considerations

This document has no IANA actions.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

10.2. Informative References

[Ayoub-16Jan]
Ayoub Benaissa, "Re: New Version Notification for draft-usama-seat-intra-vs-post-00.txt", January 2026, <https://mailarchive.ietf.org/arch/msg/seat/8eynK9ky5F-TcnL_UPbSRDKuK1E/>.

[Goran-LAKE]
G旦ran Selander, "Re: Comments for remote attestation over EDHOC", May 2024, <https://mailarchive.ietf.org/arch/msg/lake/Bb3eTcQxDA-F1AYJ0hZZy3p9wpQ/>.

[I-D.ietf-tls-rfc8446bis]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-rfc8446bis-14, 13 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-rfc8446bis-14>.

[I-D.jiang-seat-dynamic-attestation]
Jiang, Y. and Wangdonghui, "Dynamic Attestation for AI Agent Communication", Work in Progress, Internet-Draft, draft-jiang-seat-dynamic-attestation-00, 13 November 2025, <https://datatracker.ietf.org/doc/html/draft-jiang-seat-dynamic-attestation-00>.

[I-D.mihalcea-seat-use-cases]
Mihalcea, I., Sardar, M. U., Fossati, T., Reddy.K, T., Jiang, Y., and M. Chen, "Use Cases and Properties for Integrating Remote Attestation with Secure Channel Protocols", Work in Progress, Internet-Draft, draft-mihalcea-seat-use-cases-01, 19 January 2026, <https://datatracker.ietf.org/doc/html/draft-mihalcea-seat-use-cases-01>.

[ID-Crisis]
Sardar, M. U., Moustafa, M., and T. Aura, "Identity Crisis in Confidential Computing: Formal Analysis of Attested TLS", November 2025, <https://www.researchgate.net/publication/398839141_Identity_Crisis_in_Confidential_Computing_Formal_Analysis_of_Attested_TLS>.

[ID-Crisis-Repo]
Muhammad Usama Sardar, "Identity Crisis in Confidential Computing: Formal analysis of attested TLS protocols", <https://github.com/CCC-Attestation/formal-spec-id-crisis>.

[Keith-STET-CCC]
Keith Moyer, "Split-Trust Encryption Tool Attested Session Handling", March 2022, <https://github.com/CCC-Attestation/meetings/blob/main/materials/KeithMoyer_STET.pdf>.

[Markus-16Jan]
Markus Rudy, "Re: New Version Notification for draft-usama-seat-intra-vs-post-00.txt", January 2026, <https://mailarchive.ietf.org/arch/msg/seat/Pxr_12v6MIQIzGFTUdx04aVZYpM/>.

[Markus-19Jan]
Markus Rudy, "Re: New Version Notification for draft-usama-seat-intra-vs-post-00.txt", January 2026, <https://mailarchive.ietf.org/arch/msg/seat/iAeCQLna8FdfoQGV3P-mEHUobn4/>.

[MCR-LAKE] Michael Richardson, "Re: Comments for remote attestation over EDHOC", May 2024, <https://mailarchive.ietf.org/arch/msg/lake/RseQknOug41sTzW7xBJ60oRdvq0/>.

[MCR-LAKE2]
Michael Richardson, "Evaluation of attestation results by EDHOC clients", June 2024, <https://mailarchive.ietf.org/arch/msg/lake/o2oujiDacHm2m9a5y7W50oUsY7o/>.

[Mike-19Jan]
Mike Bursell, "Re: Requesting review of IETF draft on categories for attested TLS", January 2026, <https://lists.confidentialcomputing.io/g/attestation/message/276>.

[RA-TLS] Sardar, M. U., Niemi, A., Tschofenig, H., and T. Fossati, "Towards Validation of TLS 1.3 Formal Model and Vulnerabilities in Intel’s RA-TLS Protocol", November 2024, <https://www.researchgate.net/publication/385384309_Towards_Validation_of_TLS_13_Formal_Model_and_Vulnerabilities_in_Intel’s_RA-TLS_Protocol>.

[RelayAttacks]
Sardar, M. U., "Relay Attacks in Intra-handshake Attestation for Confidential Agentic AI Systems", January 2026, <https://mailarchive.ietf.org/arch/msg/seat/x3eQxFjQFJLceae6l4_NgXnmsDY/>.

[RFC9261] Sullivan, N., "Exported Authenticators in TLS", RFC 9261, DOI 10.17487/RFC9261, July 2022, <https://www.rfc-editor.org/rfc/rfc9261>.

[RFC9266] Whited, S., "Channel Bindings for TLS 1.3", RFC 9266, DOI 10.17487/RFC9266, July 2022, <https://www.rfc-editor.org/rfc/rfc9266>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/rfc/rfc9334>.

[SEAT-Charter]
IETF, "Secure Evidence and Attestation Transport (SEAT): Charter for Working Group", <https://datatracker.ietf.org/wg/seat/about/>.

[SoK-Attestation]
Sardar, M. U., Fossati, T., and S. Frost, "SoK: Attestation in Confidential Computing", January 2023, <https://www.researchgate.net/publication/367284929_SoK_Attestation_in_Confidential_Computing>.

[Stunes-vTPM-CCC]
Mike Stunes, "Azure vTPM Attestation and Binding", July 2025, <https://www.youtube.com/watch?v=J7SibeZmQsE>.

[Tech-Concepts]
Sardar, M. U., "Perspicuity of Attestation Mechanisms in Confidential Computing: Technical Concepts", October 2025, <https://www.researchgate.net/publication/396199290_Perspicuity_of_Attestation_Mechanisms_in_Confidential_Computing_Technical_Concepts>.

[Usama-TLS-26Feb25]
Muhammad Usama Sardar, "Impersonation attacks on protocol in draft-fossati-tls-attestation (Identity crisis in Attested TLS) for Confidential Computing", February 2025, <https://mailarchive.ietf.org/arch/msg/tls/Jx_yPoYWMIKaqXmPsytKZBDq23o/>.

Acknowledgments

We gratefully thank the following:

* Peg Jones for review of early draft before submission of -00

* Paul Wouters for review of section 4 of -00

* Ayoub Benaissa for review of -00 and sharing his practical experiences

* Markus Rudy for review of -00 and sharing his practical experiences and for conducting experiments with TDX and SNP on our request

* Mike Bursell (Executive Director, Confidential Computing Consortium) for review of -01 [Mike-19Jan]

Contributors

Pavel Nikonorov (GENXT / IIAP NAS RA) contributed text in Section 4.1.1 and Section 5.2.1.

History

-01

* Added scope section to address comments of Paul Wouters

* Added comments of Ayoub Benaissa as quotes

* Added some subsections to incorporate practical experiences of Markus Rudy

* Extended security considerations

-02

* Added experiments by Markus Rudy

* Removed our experiments

* Added reference to use cases document to address comment of Mike Bursell

Author’s Address

Muhammad Usama Sardar
TU Dresden
Email: muhammad_usama.sardar@tu-dresden.de