Remote ATtestation ProcedureS
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

L. Xia
W. Jiang
Huawei Technologies
M. U. Sardar
TU Dresden
H. Birkholz
Fraunhofer SIT
J. Zhang
H. Labiod
Huawei Technologies France S.A.S.U.

20 October 2025

Integration of Remote Attestation with Key Negotiation and Key Distribution mechanisms
draft-xia-rats-key-negotiation-integration-01

Abstract

This draft proposes a lightweight security enhancement scheme based on integration of remote attestation with key negotiation and key distribution. Correctly integrating the main steps of end-to-end key negotiation into the remote attestation process may provide more security and flexibility.

About This Document

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

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-xia-rats-key-negotiation-integration/.

Discussion of this document takes place on the Remote ATtestation ProcedureS Working Group mailing list (mailto:rats@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/rats/. Subscribe at https://www.ietf.org/mailman/listinfo/rats/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-rats/draft-xia-rats-key-negotiation-integration.

Status of This Memo

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

This Internet-Draft will expire on 23 April 2026.

Copyright Notice

Table of Contents

1. Introduction

Remote attestation is a security mechanism based on trusted hardware (e.g., TPM, TEE) to allow remote verifiers to cryptographically verify the integrity of the target device's software configuration, hardware state, and runtime environment. Therefore, remote attestation can effectively demonstrate the overall security status of endpoints in a communication. Secure channel protocols (e.g., TLS, QUIC, IPSec and SSH) establish end-to-end (E2E) secure channels based on the authentication of the endpoint's legitimate identity and secure key negotiation, thereby ensuring the security of network communication. Combining remote attestation protocols with secure channel protocols correctly and establishing cryptographic binding between them achieves a logical binding of endpoint security and network security. This ensures dual verification and protection of the endpoint's identity and state in secure connections. Attested TLS [I-D.fossati-tls-attestation] [I-D.fossati-tls-exported-attestation] is currently important related work in the industry. Other similar works include binding remote attestation with credential issuance (e.g., certificates [I-D.ietf-lamps-csr-attestation], OAuth tokens, etc.) to enhance security.

However, in some scenarios, the above binding may not be possible. For example:

* Scenario 1: When tenants in a public cloud/compute cluster deploy workloads, they need to first verify the security of their runtime environment through remote attestation before requesting the Key Management Service (KMS) to assign application-layer data keys to the virtual machine (VM)/compute node on which the workload is running. These keys are then used for application-layer data encryption, such as file encryption or disk encryption, rather than for any secure channel protocols. For example, to protect AI/ML model parameters from leakage and tampering, for example, model weights and other parameters must be encrypted before transmiting and loading the model to a Trusted Execution Environment (TEE) to ensure that only the TEE with the right the key can decrypts it.

* Scenario 2: The end user/client accesses the online TEE computing environment, submits their data for business processing or large model inference and must ensure the security of the entire computing environment through remote attestation before establishing a secure connection. There are multiple options for the secure channel protocol that can be established for this use

case, such as TLS, IPSec, QUIC and OHTTP. The user may only need to complete end-to-end (E2E) key negotiation based on remote attestation. The negotiated key can be used for various implementation methods, depending on which secure protocol or application layer encryption is used.

* Scenario 3: A company intends to deploy its private large model or file system in the trusted execution environment (TEE) of a public cloud platform. In order to keep the deployed content confidential, the company first uploads the encrypted objects to the cloud platform. It then conducts a security check on the cloud platform's TEE using remote attestation. Once this has been passed, the decryption key is sent to decrypt the uploaded objects inside the TEE.

In summary, this draft introduces the idea of integrating remote attestation and security mechanisms like key negotiation and key distribution to build less complex, lightweight and enhanced security schemes. More precisely, the combination of both attestation and security mechanisms brings the followings benefits::

* The key distribution of KMS or E2E key negotiation can be completed automatically based on remote attestation, thereby improving the security of key negotiation.

* The automatically negotiated keys can be flexibly applied in various ways, whether for secure protocols or application layer encryption.

* Compared to the complete and systematic implementation of Attested TLS, a more lightweight implementation can be provided.

1.1. Use of Negotiated Keys in Different Security Protocols

If the negotiated key is used for application-layer encryption, its usage is closely tied to the application and can be highly flexible. When the key is used for security protocols such as TLS and IPsec, there are various ways to integrate and utilise it within the protocol:

* TLS: The negotiated key can be used as a pre-shared key for subsequent TLS handshakes, or as an externally imported shared key for TLS hybrid key exchange [I-D.ietf-tls-hybrid-design]. Integrating all these pre-shared keys into the TLS protocol can be achieved through the hybrid authentication mode of TLS_CERT_WITH_EXTERN_PSK. This approach not only implements a mutual authentication using the pre-shared key/SK and the server certificate/attester's identity but also verifies the binding

relationship between the SK and the attester’s identity. If the server certificate obtained during the TLS negotiation cannot be correctly bound to the currently used pre-shared key, the TLS negotiation will terminate.

* IPsec: The negotiated key can be used as a pre-shared key for subsequent IKEv2 handshakes, or as Post-quantum Preshared Keys (PPKs) [RFC8784] to be used in the IPsec protocol. Similar to TLS, the IKEv2 hybrid authentication mode using pre-shared keys and certificates achieves mutual authentication and ensures the correctness of the binding relationship between the two.

1.2. Requirements Notation

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.

2. Integration Scheme

The current specification is based on passport model of RATS. Future versions of specification will include the background-check model.

2.1. Public Cloud KMS Key Distribution Integrated with Remote Attestation

The Key Management Service (KMS) for public cloud networks includes the root of trust, secure channels between Attester (node, VM, container, service, application) and the KMS, full lifecycle management of keys (including key generation, storage, rotation, and destruction), hierarchical encryption architecture (such as Envelope Encryption), and access control mechanisms. Specifically:

* The basic process for symmetric key distribution and usage is as follows: When an application requires data to be encrypted and shared, it requests the KMS to distribute the DEK (Data Encryption Key) and the EDEK (Encrypted DEK, encrypted using the Customer Master Key (CMK)) to the application via an API. The application then encrypts the data using the DEK, deletes the DEK from memory, and sends the encrypted ciphertext along with the EDEK to the receiving application. The receiving application then requests the KMS to decrypt the EDEK via an API, retrieves the DEK to decrypt the data, and then deletes the DEK from memory.

* The basic process of asymmetric key distribution and usage is as follows: When an application needs to encrypt data, it requests the KMS to generate a public-private key pair via an API, and then uses the obtained public key to encrypt the data. The encrypted ciphertext is then sent to the receiving application. The receiving application calls the KMS's decryption API and the KMS uses the corresponding private key internally to decrypt the data and return the plaintext to the receiving application. The private key never leaves the KMS.When an application requires digital signing of data, it requests the KMS to generate a public窶冤苑rivate key pair via an API. The public key is then distributed to all parties that need to verify the signature. The application then requests the KMS to sign the data using the corresponding private key via an API, and sends the signature result along with the data to the receiving application. The receiving application uses the public key to verify the signature.

In summary, KMS provides the applications with the required deliverables, which include application-layer encryption/ authentication, symmetric/asymmetric keys, decrypted plaintext and calculated signatures. For simplicity, the term 'keys' is used here to refer to all these deliverables.

The following diagram shows the method of integrating key distribution into the remote attestation interaction Passport Model process:

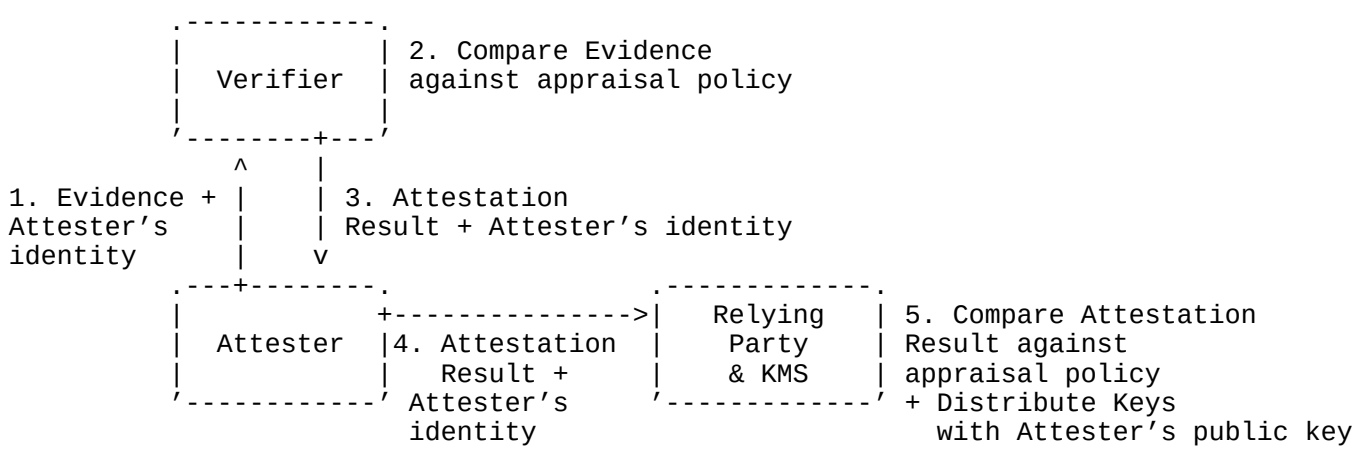


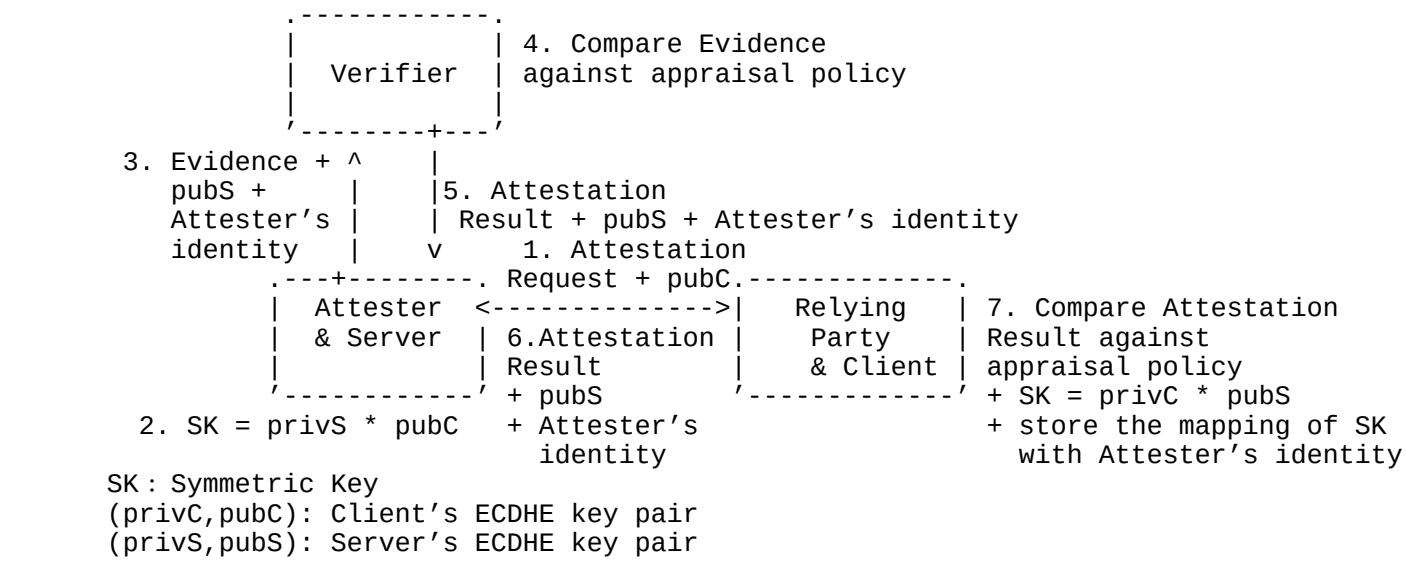
Figure 1: Public Cloud KMS Key Distribution Integrated Scheme on Passport Model of RATS

In the standard remote attestation process described above, the Attester, which is an application in TEE, can request the Attester's application layer keys after providing its attestation result to the KMS. By including the attester's identity (raw public key or certificate) in the messages throughout the remote attestation process and having the attester (using its attestation evidence signing key) and verifier (using its attestation result signing key) endorse and sign it, a key binding mechanism between the attester's attestation result and its identity is implemented. Subsequently, the KMS can use the identity's public key for key distribution, ensuring that the keys are distributed to the correct attester, thereby eliminating the risk of diversion attacks. During key rotation, the KMS can proactively trigger this process to update and rotate the new and old keys.

Overall, the above approach integrates end-to-end key distribution correctly into the remote attestation process, achieving automation of key distribution and higher security guarantees based on the security of attester endpoint.

2.2. Integrating E2E Key Negotiation Into Remote Attestation

The current main implementation mechanism for E2E key negotiation is Diffie-Hellman Ephemeral (DHE) and Elliptic-Curve Diffie-Hellman Ephemeral (ECDHE). Taking ECDHE as an example, the method of integrating it into remote attestation passport Model process is shown in the following figure:



SK : Symmetric Key
(privC,pubC): Client's ECDHE key pair
(privS,pubS): Server's ECDHE key pair

Figure 2: Integrating E2E Key Negotiation Into Remote Attestation Passport Model Interaction

In the standard remote attestation process described above,the Client includes the public key (pubC) from its dynamically generated ECDHE key pair (privC, pubC) in the remote attestation request message while retaining the private key (privC). Upon receiving pubC, the Server can compute the symmetric key SK using its private key privS from its dynamically generated ECDHE key pair (privS, pubS). At the same time, by including the Attester's identity (raw public key or certificate) in all subsequent messages of the remote attestation process and having the Attester (using its attestation evidence signing key) and Verifier (using its attestation result signing key) endorse and sign it, a key binding mechanism between the Attester's attestation result and its identity is implemented, thereby eliminating the risk of diversion attacks. After completing the remote attestation with the Verifier, the Server includes its pubS in the attestation result returned to the Client. Once the Client has verified the attestation result, it can compute the symmetric key SK using pubS and its own private key privC, thereby completing both the remote attestation and the ECDHE key agreement. The Client can also record the relationship between the generated SK and the corresponding Attester's identity. This ensures that the SK is subsequently used to establish a secure protocol connection with an Attester possessing the correct identity. Some of the metadata negotiated alongside the key exchange materials includes the session ID, nonce and algorithm.

Overall, the above approach integrates end-to-end key negotiation correctly into the remote attestation process, achieving automation of key negotiation and higher security guarantees based on the security of attester endpoint.

2.3. Enterprise Customer Key Distribution to Public Cloud Into Remote Attestation

Enterprises need to deploy data assets, such as data, applications and systems, on public clouds. Some of these assets are critical to the enterprise, such as large private model applications. It is therefore essential to ensure the trustworthiness and security of the operating environment. The process involves first uploading the encrypted data assets to the cloud environment and then performing remote attestation on the operating environment. Once the operating environment passes the security verification, the decryption key is distributed to it for loading and running the decrypted data assets. In fact, the key here can also be generalized to refer to the decrypted plaintext and the calculated signatures provided by Enterprise KMS. This scenario is similar to the public cloud KMS key

distribution scenario, except that the public cloud is no longer responsible for key distribution. Instead, the enterprise manages the keys itself and completes the key distribution after passing the security verification through remote attestation. The method is illustrated in the following diagram:

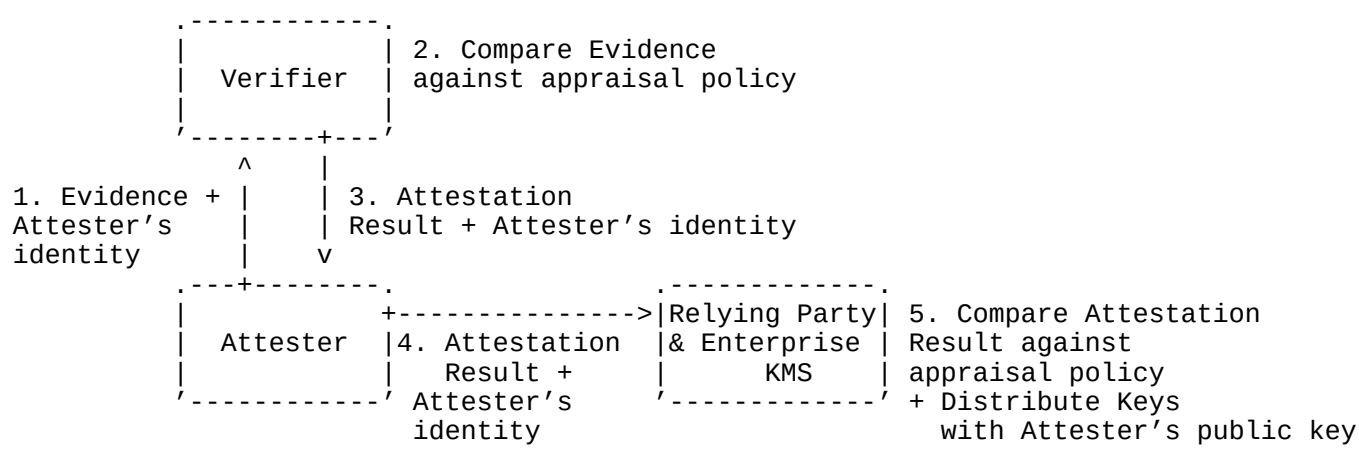


Figure 3: Enterprise KMS Key Distribution Integrated with Remote Attestation Passport Model Interaction

By including the Attester's identity (raw public key or certificate) in the messages throughout the remote attestation process and having the Attester (using its attestation evidence signing key) and Verifier (using its attestation result signing key) endorse and sign it, a key binding mechanism between the attester's attestation result and its identity is implemented. Subsequently, the enterprise KMS can use the identity's public key for key distribution, ensuring that the keys are distributed to the correct attester, thereby eliminating the risk of diversion attacks. During key rotation, the enterprise KMS can proactively trigger this process to update and rotate of the new and old keys.

Overall, the above approach integrates end-to-end key distribution correctly into the remote attestation process, achieving automation of key distribution and higher security guarantees based on the security of attester endpoint.

3. Remote Attestation Protocol and Message Extensions

This section describes how to extend RATS protocol and message to incorporate key negotiation into the remote attestation process.

TBD

4. Implementation Status

// RFC Editor: please remove this section prior to publication. This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groupsto assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

4.1. Trustee

Responsible Organisation: Trustee (open source project within the Confidential Containers).

Location: https://github.com/confidential-containers/trustee

Description: Trustee contains tools and components for attesting confidential guests and providing secrets to them. Collectively, these components are known as Trustee. Trustee typically operates on behalf of the guest owner and interact remotely with guest components. Trustee components include: - Key Broker Service: The KBS is a server that facilitates remote attestation and secret delivery. Its role is similar to that of the Relying Party in the RATS model; - Attestation Service: The AS verifies TEE evidence. In the RATS model this is a Verifier; - Reference Value Provider Service: The RVPS manages reference values used to verify TEE evidence; - KBS Client Tool: This is a simple tool which can be used to test or configure the KBS and AS.

There are two main ways to deploy Trustee: with Docker Compose, on Kubernetes.

Level of Maturity: This is a proof-of-concept prototype implementation.

License: Apache-2.0.

Coverage: This implementation covers most of the aspects of the use case 1 and 3 of this draft.

Contact: Ding Ma, xynnn@linux.alibaba.com

5. Security Considerations

Evidence should be cryptographically bound to the identifier provided to the machine by the infrastructure provider to prevent diversion attacks [Meeting-122-TLS-Slides].

6. Privacy Considerations

TBD

7. Optimization Considerations

For the first time connection between the Attester and the Relying Party, embedding the raw public key/certificate inside the Attestation Result is necessary as it simplifies the procedure to delivery of the raw pulbic key/cerficate from the Attester. For the latter connection between this Atterster and the Relying Party, if the raw public key/certificate remains unchanged, the Attester May choose to use the hash of its raw public key/cerficate instead for the Evidence/Attestation Result between itself and the Verifier, and only forward the Attestation Result that contains the hash of hash of its raw public key/cerficate. At the Relying Party side, it compares this hash with the hash of the local cached raw public key/cerficate, and use the corresponding raw public key/cerficate for this session when it matches. This will reduce the payload carried by the Evidence and the Attestation Result.

8. IANA Considerations

TBD

9. References

9.1. 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>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[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>.

[RFC8784] Fluhrer, S., Kampanakis, P., McGrew, D., and V. Smyslov, "Mixing Preshared Keys in the Internet Key Exchange Protocol Version 2 (IKEv2) for Post-quantum Security", RFC 8784, DOI 10.17487/RFC8784, June 2020, <https://www.rfc-editor.org/rfc/rfc8784>.

9.2. Informative References

[I-D.fossati-tls-attestation] Tschofenig, H., Sheffer, Y., Howard, P., Mihalcea, I., Deshpande, Y., Niemi, A., and T. Fossati, "Using Attestation in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", Work in Progress, Internet-Draft, draft-fossati-tls-attestation-09, 30 April 2025, <https://datatracker.ietf.org/doc/html/draft-fossati-tls-attestation-09>.

[I-D.fossati-tls-exported-attestation] Fossati, T., Sardar, M. U., Reddy.K, T., Sheffer, Y., Tschofenig, H., and I. Mihalcea, "Remote Attestation with Exported Authenticators", Work in Progress, Internet-Draft, draft-fossati-tls-exported-attestation-02, 3 July 2025, <https://datatracker.ietf.org/doc/html/draft-fossati-tls-exported-attestation-02>.

[I-D.ietf-lamps-csr-attestation] Ounsworth, M., Tschofenig, H., Birkholz, H., Wiseman, M., and N. Smith, "Use of Remote Attestation with Certification Signing Requests", Work in Progress, Internet-Draft, draft-ietf-lamps-csr-attestation-21, 5 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-lamps-csr-attestation-21>.

[I-D.ietf-tls-hybrid-design] Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid key exchange in TLS 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-hybrid-design-16, 7 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-tls-hybrid-design-16>.

[Meeting-122-TLS-Slides]
Sardar, M. U., Moustafa, M., and T. Aura, "Identity Crisis in Attested TLS for Confidential Computing", March 2025, <https://datatracker.ietf.org/meeting/122/materials/slides-122-tls-identity-crisis-00>.

Authors' Addresses

Liang Xia
Huawei Technologies
Email: frank.xialiang@huawei.com

Weiyu Jiang
Huawei Technologies
Email: jiangweiyu1@huawei.com

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

Henk Birkholz
Fraunhofer SIT
Email: henk.birkholz@ietf.contact

Jun Zhang
Huawei Technologies France S.A.S.U.
Email: junzhang1@huawei.com

Houda Labiod
Huawei Technologies France S.A.S.U.
Email: houda.labiod@huawei.com