Workgroup: TLS
Internet-Draft: draft-ietf-tls-dtls-rrc-06
Updates: 6347, 9147 (if approved)
Published: 6 July 2022
Intended Status: Standards Track
Expires: 7 January 2023
H. Tschofenig, Ed.
Arm Limited
A. Kraus
T. Fossati
Arm Limited

Return Routability Check for DTLS 1.2 and DTLS 1.3

Abstract

This document specifies a return routability check for use in context of the Connection ID (CID) construct for the Datagram Transport Layer Security (DTLS) protocol versions 1.2 and 1.3.

Discussion Venues

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

Discussion of this document takes place on the Transport Layer Security Working Group mailing list (tls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/tls/.

Source for this draft and an issue tracker can be found at https://github.com/tlswg/dtls-rrc.

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/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 7 January 2023.

Copyright Notice

Table of Contents

1. Introduction

In "classical" DTLS, selecting a security context of an incoming DTLS record is accomplished with the help of the 5-tuple, i.e. source IP address, source port, transport protocol, destination IP address, and destination port. Changes to this 5 tuple can happen for a variety reasons over the lifetime of the DTLS session. In the IoT context, NAT rebinding is common with sleepy devices. Other examples include end host mobility and multi-homing. Without CID, if the source IP address and/or source port changes during the lifetime of an ongoing DTLS session then the receiver will be unable to locate the correct security context. As a result, the DTLS handshake has to be re-run. Of course, it is not necessary to re-run the full handshake if session resumption is supported and negotiated.

A CID is an identifier carried in the record layer header of a DTLS datagram that gives the receiver additional information for selecting the appropriate security context. The CID mechanism has been specified in [RFC9146] for DTLS 1.2 and in [RFC9147] for DTLS 1.3.

Section 6 of [RFC9146] describes how the use of CID increases the attack surface by providing both on-path and off-path attackers an opportunity for (D)DoS. It then goes on describing the steps a DTLS principal must take when a record with a CID is received that has a source address (and/or port) different from the one currently associated with the DTLS connection. However, the actual mechanism for ensuring that the new peer address is willing to receive and process DTLS records is left open. This document standardizes a return routability check (RRC) as part of the DTLS protocol itself.

The return routability check is performed by the receiving peer before the CID-to-IP address/port binding is updated in that peer's session state database. This is done in order to provide more confidence to the receiving peer that the sending peer is reachable at the indicated address and port.

Note however that, irrespective of CID, if RRC has been successfully negotiated by the peers, path validation can be used at any time by either endpoint. For instance, an endpoint might use RRC to check that a peer is still in possession of its address after a period of quiescence.

2. Conventions and Terminology

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.

This document assumes familiarity with the CID format and protocol defined for DTLS 1.2 [RFC9146] and for DTLS 1.3 [RFC9147]. The presentation language used in this document is described in Section 4 of [RFC8446].

This document reuses the definition of "anti-amplification limit" from [RFC9000] to mean three times the amount of data received from an unvalidated address. This includes all DTLS records originating from that source address, excluding discarded ones.

3. RRC Extension

The use of RRC is negotiated via the rrc DTLS-only extension. On connecting, the client includes the rrc extension in its ClientHello if it wishes to use RRC. If the server is capable of meeting this requirement, it responds with a rrc extension in its ServerHello. The extension_type value for this extension is TBD1 and the extension_data field of this extension is empty. The client and server **MUST NOT** use RRC unless both sides have successfully exchanged rrc extensions.

Note that the RRC extension applies to both DTLS 1.2 and DTLS 1.3.

4. Return Routability Check Message Types

This document defines the return_routability_check content type (Figure 1) to carry Return Routability Check protocol messages.

The protocol consists of three message types: path_challenge, path_response and path_drop that are used for path validation and selection as described in Section 7.

Each message carries a Cookie, a 8-byte field containing arbitrary data.

The return_routability_check message **MUST** be authenticated and encrypted using the currently active security context.

```
enum {
    invalid(0),
    change_cipher_spec(20),
    alert(21),
    handshake(22),
    application_data(23),
    heartbeat(24),  /* RFC 6520 */
    tls12_cid(25),  /* RFC 9146, DTLS 1.2 only */
    return_routability_check(TBD2), /* NEW */
    (255)
} ContentType;

uint64 Cookie;

enum {
    path_challenge(0),
    path_response(1),
    path_drop(2),
    (255)
} rrc_msg_type;

struct {
    rrc_msg_type msg_type;
    select (return_routability_check.msg_type) {
        case path_challenge: Cookie;
        case path_response:  Cookie;
        case path_drop:      Cookie;
    };
} return_routability_check;
```

Figure 1: Return Routability Check Message

Future extensions or additions to the Return Routability Check protocol may define new message types. Implementations **MUST** be able to parse and ignore messages with an unknown msg_type.

5. RRC and CID Interplay

RRC offers an in-protocol mechanism to perform peer address validation that complements the "peer address update" procedure described in Section 6 of [RFC9146]. Specifically, when both CID [RFC9146] and RRC have been successfully negotiated for the session, if a record with CID is received that has the source address of the enclosing UDP datagram different from the one currently associated with that CID value, the receiver **SHOULD** perform a return routability

check as described in Section 7, unless an application layer specific address validation mechanism can be triggered instead.

6. Attacker Model

We define two classes of attackers, off-path and on-path, with increasing capabilities (see Figure 2) partly following terminology introduced in QUIC [RFC9000]:

- An off-path attacker is not on the original path between the DTLS peers, but is able to observe packets on the original path and has faster routing compared to the DTLS peers, which allows it to make copies of the observed packets, race its copies to either peer and consistently win the race.

- An on-path attacker is on the original path between the DTLS peers and is therefore capable, compared to the off-path attacker, to also drop and delay records at will.

Note that in general, attackers cannot craft DTLS records in a way that would successfully pass verification due to the cryptographic protections applied by the DTLS record layer.

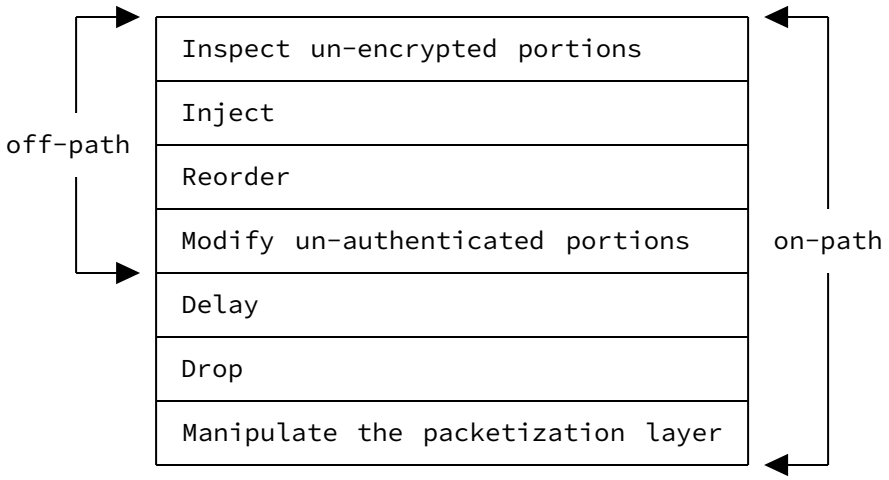


Figure 2: Attackers capabilities

RRC is designed to defend against the following attacks:

- On-path and off-path attackers that try to create an amplification attack by spoofing the source address of the victim (Section 6.1).

- Off-path attackers that try to put themselves on-path (Section 6.2), provided that the enhanced path validation algorithm is used (Section 7.2).

6.1. Amplification

Both on-path and off-path attackers can send a packet (either by modifying it on the fly, or by copying, injecting and racing it, respectively) with the source address modified to that of a victim

host. If the traffic generated by the server in response is larger compared to the received packet (e.g., a CoAP server returning an MTU's worth of data from a 20-bytes GET request) the attacker can use the server as a traffic amplifier toward the victim.

When receiving a packet with a known CID and a spoofed source address, an RRC-capable endpoint will not send a (potentially large) response but instead a small path_challenge message to the victim host. Since the host is not able to decrypt it and generate a valid path_response, the address validation fails, which in turn keeps the original address binding unaltered.

Note that in case of an off-path attacker, the original packet still reaches the intended destination; therefore, an implementation could use a different strategy to mitigate the attack.

6.2. Off-Path Packet Forwarding

An off-path attacker that can observe packets might forward copies of genuine packets to endpoints over a different path. If the copied packet arrives before the genuine packet, this will appear as a path change, like in a genuine NAT rebinding occurrence. Any genuine packet will be discarded as a duplicate. If the attacker is able to continue forwarding packets, it might be able to cause migration to a path via the attacker. This places the attacker on-path, giving it the ability to observe or drop all subsequent packets.

This style of attack relies on the attacker using a path that has the same or better characteristics (e.g., due to a more favourable service level agreements) as the direct path between endpoints. The attack is more reliable if relatively few packets are sent or if packet loss coincides with the attempted attack.

A data packet received on the original path that increases the maximum received packet number will cause the endpoint to move back to that path. Therefore, eliciting packets on this path increases the likelihood that the attack is unsuccessful. Note however that, unlike QUIC, DTLS has no "non-probing" packets so this would require application specific mechanisms.

Figure 3 illustrates the case where a receiver receives a packet with a new source IP address and/or new port number. In order to determine whether this path change was not triggered by an off-path attacker, the receiver will send a RRC message of type path_challenge (1) on the old path.

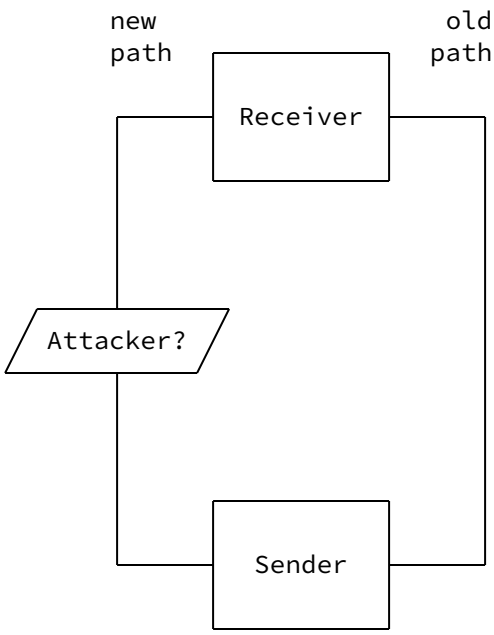


Figure 3: Off-Path Packet Forwarding Scenario

Three cases need to be considered:

Case 1: The old path is dead (e.g., due to a NAT rebinding), which leads to a timeout of (1).

As shown in Figure 4, a path_challenge (2) needs to be sent on the new path. If the sender replies with a path_response on the new path (3), the switch to the new path is considered legitimate.

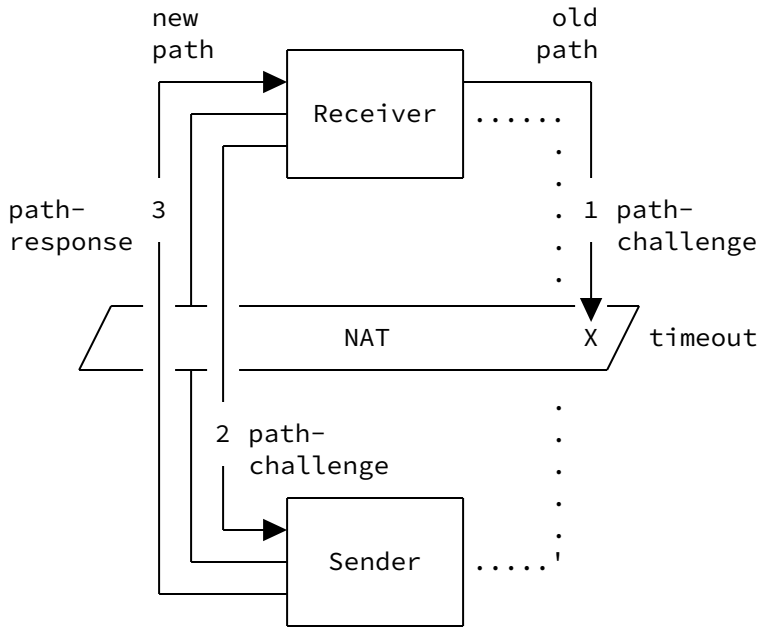


Figure 4: Old path is dead

Case 2: The old path is alive but not preferred.

This case is shown in Figure 5 whereby the sender replies with a path_drop message (2) on the old path. This triggers the receiver to send a path_challenge (3) on the new path. The sender will reply with a path_response (4) on the new path, thus providing confirmation for the path migration.

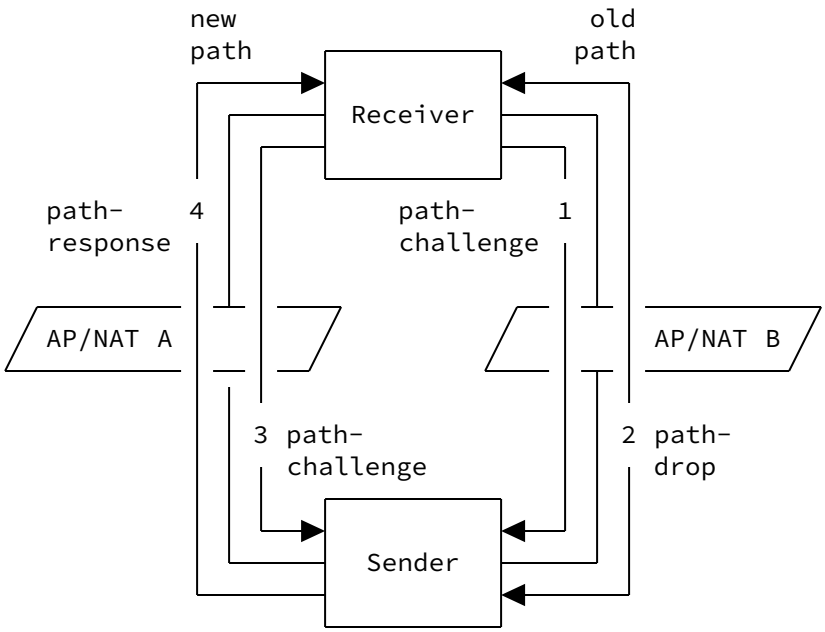


Figure 5: Old path is not preferred

Case 3: The old path is alive and preferred.

This is most likely the result of an off-path attacker trying to place itself on path. The receiver sends a path_challenge on the old path and the sender replies with a path_response (2) on the old path. The interaction is shown in Figure 6. This results in the connection not being migrated to the new path, thus thwarting the attack.

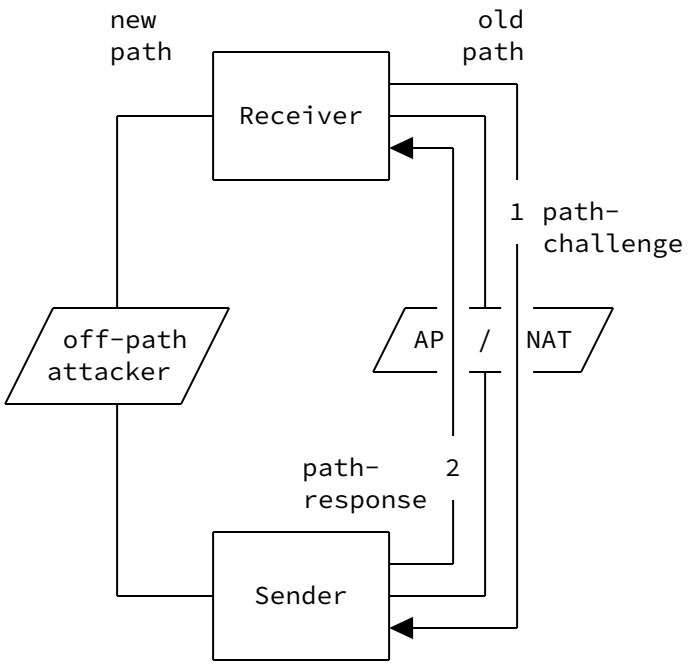


Figure 6: Old path is preferred

Note that this defense is imperfect, but this is not considered a serious problem. If the path via the attack is reliably faster than the old path despite multiple attempts to use that old path, it is not possible to distinguish between an attack and an improvement in routing.

An endpoint could also use heuristics to improve detection of this style of attack. For instance, NAT rebinding is improbable if packets were recently received on the old path; similarly, rebinding is rare on IPv6 paths. Endpoints can also look for duplicated packets. Conversely, a change in connection ID is more likely to indicate an intentional migration rather than an attack. Note that changes in connection IDs are supported in DTLS 1.3 but not in DTLS 1.2.

7. Path Validation Procedure

The receiver that observes the peer's address or port update **MUST** stop sending any buffered application data, or limit the data sent to the unvalidated address to the anti-amplification limit.

It then initiates the return routability check that proceeds as described either in Section 7.2 or Section 7.1, depending on whether the off-path attacker scenario described in Section 6.2 is to be taken into account or not.

(The decision on what strategy to choose depends mainly on the threat model, but may also be influenced by other considerations. Examples of impacting factors include: the need to minimise implementation complexity, privacy concerns, the need to reduce the time it takes to switch path. The choice may be offered as a configuration option to the user.)

After the path validation procedure is completed, any pending send operation is resumed to the bound peer address.

Section 7.3 and Section 7.4 list the requirements for the initiator and responder roles, broken down per protocol phase.

7.1. Basic

1. The receiver creates a return_routability_check message of type path_challenge and places the unpredictable cookie into the message.

2. The message is sent to the observed new address and a timer T (see Section 7.5) is started.

3. The peer endpoint cryptographically verifies the received return_routability_check message of type path_challenge and responds by echoing the cookie value in a return_routability_check message of type path_response.

4. When the initiator receives the return_routability_check message of type path_response and verifies that it contains the sent cookie, it updates the peer address binding.

5. If T expires the peer address binding is not updated.

7.2. Enhanced

1. The receiver creates a return_routability_check message of type path_challenge and places the unpredictable cookie into the message.

2. The message is sent to the previously valid address, which corresponds to the old path. Additionally, a timer T, see Section 7.5, is started.

3. If the path is still functional, the peer endpoint cryptographically verifies the received return_routability_check message of type path_challenge. The action to be taken depends on whether the path through which the message was received is the preferred one or not anymore:

   * If the path through which the message was received is preferred, a return_routability_check message of type path_response **MUST** be returned.

*If the path through which the message was received is not preferred, a return_routability_check message of type path_drop **MUST** be returned. In either case, the peer endpoint echoes the cookie value in the response.

4. The initiator receives and verifies that the return_routability_check message contains the previously sent cookie. The actions taken by the initiator differ based on the received message:

   *When a return_routability_check message of type path_response was received, the initiator **MUST** continue using the previously valid address, i.e. no switch to the new path takes place and the peer address binding is not updated.

   *When a return_routability_check message of type path_drop was received, the initiator **MUST** perform a return routability check on the observed new address, as described in Section 7.1.

5. If T expires the peer address binding is not updated. In this case, the initiator **MUST** perform a return routability check on the observed new address, as described in Section 7.1.

7.3. Path Challenge Requirements

*The initiator **MAY** send multiple return_routability_check messages of type path_challenge to cater for packet loss on the probed path.

   -Each path_challenge **SHOULD** go into different transport packets. (Note that the DTLS implementation may not have control over the packetization done by the transport layer.)

   -The transmission of subsequent path_challenge messages **SHOULD** be paced to decrease the chance of loss.

   -Each path_challenge message **MUST** contain random data.

*The initiator **MAY** use padding using the record padding mechanism available in DTLS 1.3 (and in DTLS 1.2, when CID is enabled on the sending direction) up to the anti-amplification limit to probe if the path MTU (PMTU) for the new path is still acceptable.

7.4. Path Response/Drop Requirements

*The responder **MUST NOT** delay sending an elicited path_response or path_drop messages.

*The responder **MUST** send exactly one path_response or path_drop message for each received path_challenge.

*The responder **MUST** send the path_response or the path_drop on the path where the corresponding path_challenge has been received, so that validation succeeds only if the path is functional in both directions. The initiator **MUST NOT** enforce this behaviour.

*The initiator **MUST** silently discard any invalid path_response or path_drop it receives.

Note that RRC does not cater for PMTU discovery on the reverse path. If the responder wants to do PMTU discovery using RRC, it should initiate a new path validation procedure.

7.5. Timer Choice

When setting T, implementations are cautioned that the new path could have a longer round-trip time (RTT) than the original.

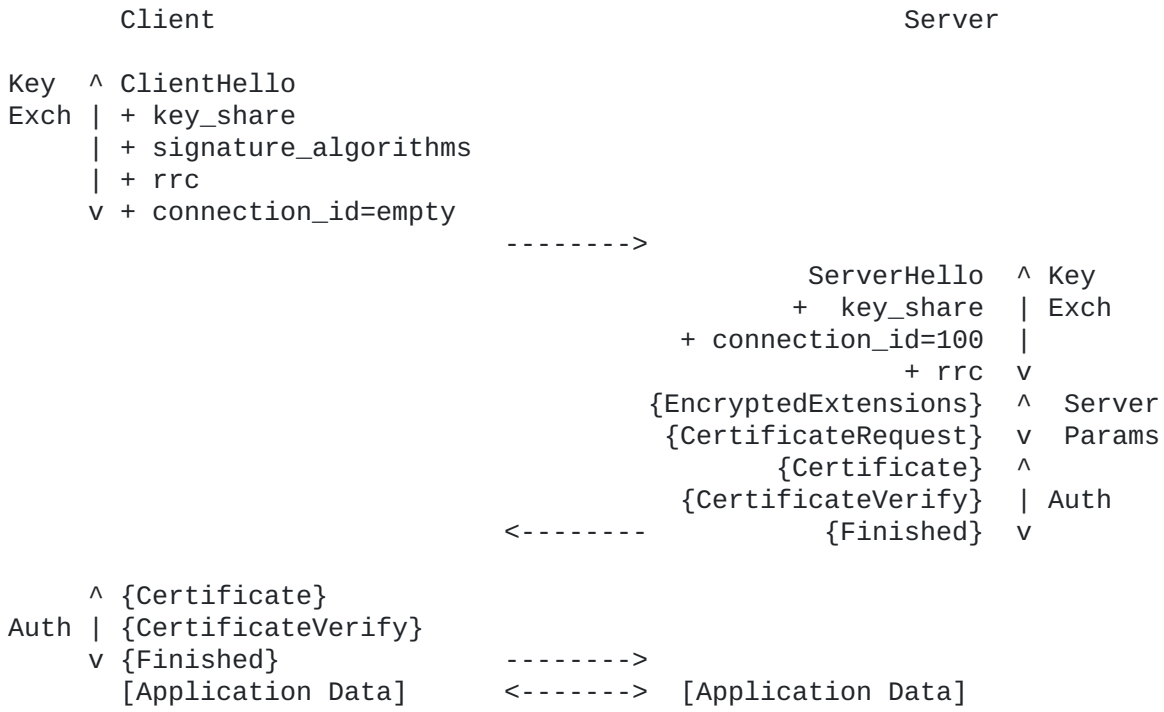
In settings where there is external information about the RTT of the active path, implementations **SHOULD** use T = 3xRTT.

If an implementation has no way to obtain information regarding the RTT of the active path, a value of 1s **SHOULD** be used.

Profiles for specific deployment environments -- for example, constrained networks [I-D.ietf-uta-tls13-iot-profile] -- **MAY** specify a different, more suitable value.

8. Example

The example TLS 1.3 handshake shown in Figure 7 shows a client and a server negotiating the support for CID and for the RRC extension.



\+ Indicates noteworthy extensions sent in the previously noted message.

* Indicates optional or situation-dependent messages/extensions that are not always sent.

{} Indicates messages protected using keys derived from a [sender]_handshake_traffic_secret.

[] Indicates messages protected using keys derived from [sender]_application_traffic_secret_N.

Figure 7: Message Flow for Full TLS Handshake

Once a connection has been established the client and the server exchange application payloads protected by DTLS with an unilaterally used CIDs. In our case, the client is requested to use CID 100 for records sent to the server.

At some point in the communication interaction the IP address used by the client changes and, thanks to the CID usage, the security context to interpret the record is successfully located by the server. However, the server wants to test the reachability of the client at his new IP address.

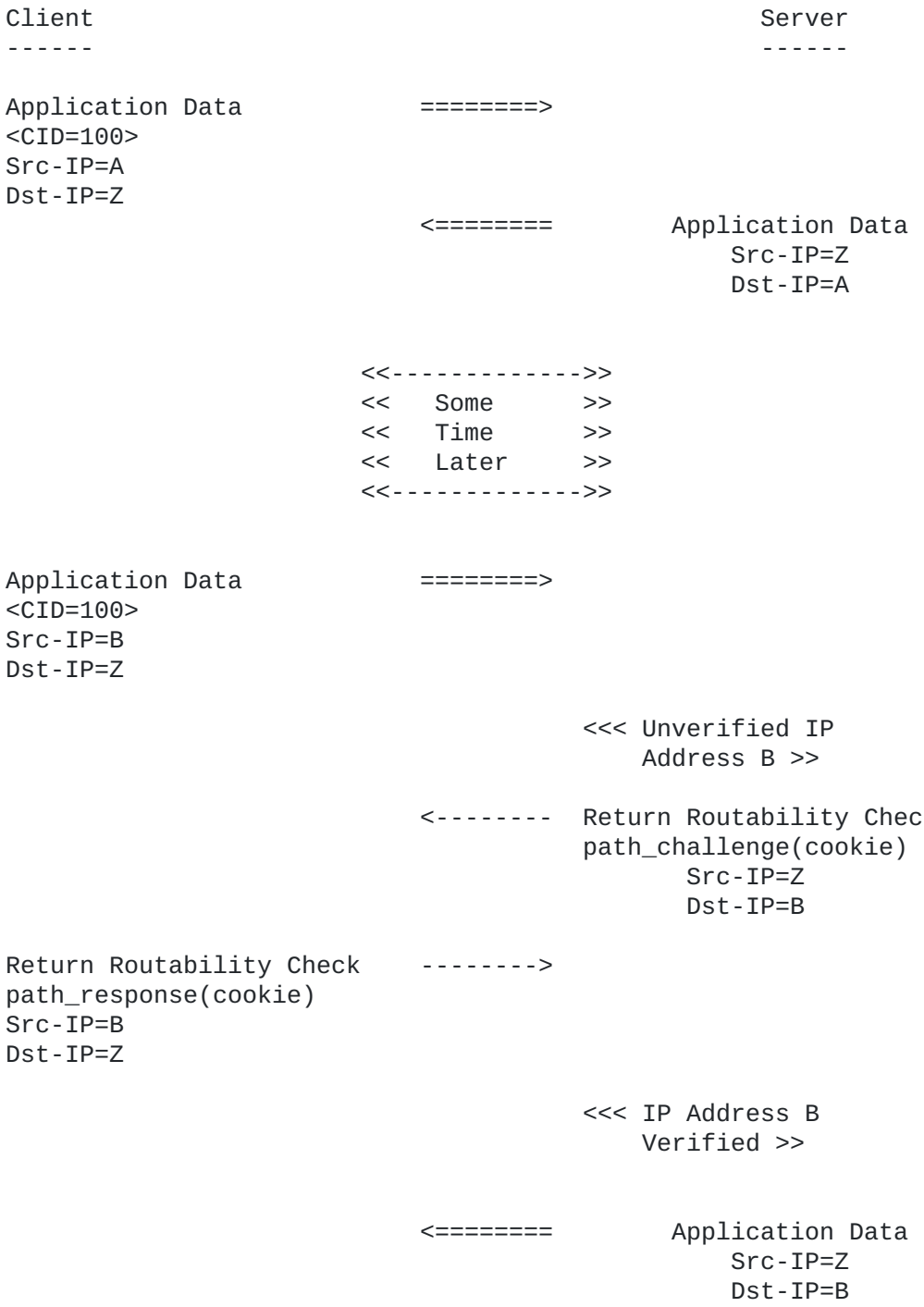


Figure 8: Return Routability Example

9. Security and Privacy Considerations

Note that the return routability checks do not protect against flooding of third-parties if the attacker is on-path, as the attacker can redirect the return routability checks to the real peer (even if those datagrams are cryptographically authenticated). On-path adversaries can, in general, pose a harm to connectivity.

When using DTLS 1.3, peers **SHOULD** avoid using the same CID on multiple network paths, in particular when initiating connection

migration or when probing a new network path, as described in Section 7, as an adversary can otherwise correlate the communication interaction across those different paths. DTLS 1.3 provides mechanisms to ensure that a new CID can always be used. In general, an endpoint should proactively send a RequestConnectionId message to ask for new CIDs as soon as the pool of spare CIDs is depleted (or goes below a threshold). Also, in case a peer might have exhausted available CIDs, a migrating endpoint could include NewConnectionId in packets sent on the new path to make sure that the subsequent path validation can use fresh CIDs.

Note that DTLS 1.2 does not offer the ability to request new CIDs during the session lifetime since CIDs have the same life-span of the connection. Therefore, deployments that use DTLS in multihoming environments **SHOULD** refuse to use CIDs with DTLS 1.2 and switch to DTLS 1.3 if the correlation privacy threat is a concern.

10. IANA Considerations

RFC Editor: please replace RFCthis with this RFC number and remove this note.

10.1. New TLS ContentType

IANA is requested to allocate an entry to the TLS ContentType registry, for the return_routability_check(TBD2) message defined in this document. The return_routability_check content type is only applicable to DTLS 1.2 and 1.3.

10.2. New TLS ExtensionType

IANA is requested to allocate the extension code point (TBD1) for the rrc extension to the TLS ExtensionType Values registry as described in Table 1.

Value	Extension Name	TLS 1.3	DTLS-Only	Recommended	Reference
TBD1	rrc	CH, SH	Y	N	RFCthis

Table 1: rrc entry in the TLS ExtensionType Values registry

10.3. New RRC Message Type Sub-registry

IANA is requested to create a new sub-registry for RRC Message Types in the TLS Parameters registry [IANA.tls-parameters], with the policy "expert review" [RFC8126].

Each entry in the registry must include:

Value: A number in the range from 0 to 255 (decimal)

Description: a brief description of the message

DTLS-Only: RRC is only available in DTLS, therefore this column will be set to Y for all the entries in this registry

Reference:
a reference document

The initial state of this sub-registry is as follows:

Value	Description	DTLS-Only	Reference
0	path_challenge	Y	RFCthis
1	path_response	Y	RFCthis
2	path_drop	Y	RFCthis
3-255	Unassigned		

Table 2: Initial Entries in RRC Message Type registry

11. Open Issues

Issues against this document are tracked at https://github.com/tlswg/dtls-rrc/issues

12. Acknowledgments

We would like to thank Achim Kraus, Hanno Becker, Hanno Boeck, Manuel Pegourie-Gonnard, Mohit Sahni and Rich Salz for their input to this document.

13. References

13.1. Normative References

[IANA.tls-parameters] IANA, "Transport Layer Security (TLS) Parameters", <https://www.iana.org/assignments/tls-parameters>.

[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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9146] Rescorla, E., Ed., Tschofenig, H., Ed., Fossati, T., and A. Kraus, "Connection Identifier for DTLS 1.2", RFC 9146, DOI 10.17487/RFC9146, March 2022, <https://www.rfc-editor.org/rfc/rfc9146>.

[RFC9147] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", RFC 9147, DOI 10.17487/RFC9147, April 2022, <https://www.rfc-editor.org/rfc/rfc9147>.

13.2. Informative References

[I-D.ietf-uta-tls13-iot-profile] Tschofenig, H. and T. Fossati, "TLS/DTLS 1.3 Profiles for the Internet of Things", Work in Progress, Internet-Draft, draft-ietf-uta-tls13-iot-profile-04, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-uta-tls13-iot-profile-04>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

Appendix A. History

RFC EDITOR: PLEASE REMOVE THIS SECTION

draft-ietf-tls-dtls-rrc-06

- *Add Achim as co-author
- *Added IANA registry for RRC message types (#14)
- *Small fix in the path validation algorithm (#15)
- *Renamed path_delete to path_drop (#16)
- *Added an "attacker model" section (#17, #31, #44, #45, #48)
- *Add criteria for choosing between basic and enhanced path validation (#18)
- *Reorganise Section 4 a bit (#19)
- *Small fix in Path Response/Drop Requirements section (#20)
- *Add privacy considerations wrt CID reuse (#30)

draft-ietf-tls-dtls-rrc-05

- *Added text about off-path packet forwarding

draft-ietf-tls-dtls-rrc-04

- *Re-submitted draft to fix references

draft-ietf-tls-dtls-rrc-03

- *Added details for challenge-response exchange

draft-ietf-tls-dtls-rrc-02

*Undo the TLS flags extension for negotiating RRC, use a new extension type

draft-ietf-tls-dtls-rrc-01

*Use the TLS flags extension for negotiating RRC

*Enhanced IANA consideration section

*Expanded example section

*Revamp message layout:

-Use 8-byte fixed size cookies

-Explicitly separate path challenge from response

draft-ietf-tls-dtls-rrc-00

*Draft name changed after WG adoption

draft-tschofenig-tls-dtls-rrc-01

*Removed text that overlapped with draft-ietf-tls-dtls-connection-id

draft-tschofenig-tls-dtls-rrc-00

*Initial version

Authors' Addresses

Hannes Tschofenig (editor)
Arm Limited

Email: hannes.tschofenig@arm.com

Achim Kraus

Email: achimkraus@gmx.net

Thomas Fossati
Arm Limited

Email: thomas.fossati@arm.com