Trust Router Protocol Flows with No Established Keys

When there are no existing keys anywhere in the Trust Router mediated ecosystem, then the protocol flow that enables an RP and an IdP to securely establish a connection is a simple 46 step process (!) that recursively calls trust router that ultimately ends up with both the RP and IdP, and the IdP and APC, having a shared set of keys each to be used for secure communications. This page details this protocol flow.

Contents

- 1. Overview
- 2. Detail

1. Overview

Understanding the subtleties of how Trust Router bootstraps itself when no shared keys exist is not easy; it will take effort, concentration, and probably an hour or two of whiteboarding going through each step one at a time. Jumping straight in to this worst case is probably not the best way to understand it. Instead, understand the simple case first - Trust Router Protocol Flows with Existing RP/IdP Key, then the intermediate case - Trust Router Protocol Flows with Existing APC/IdP Key, and then finally come back here.

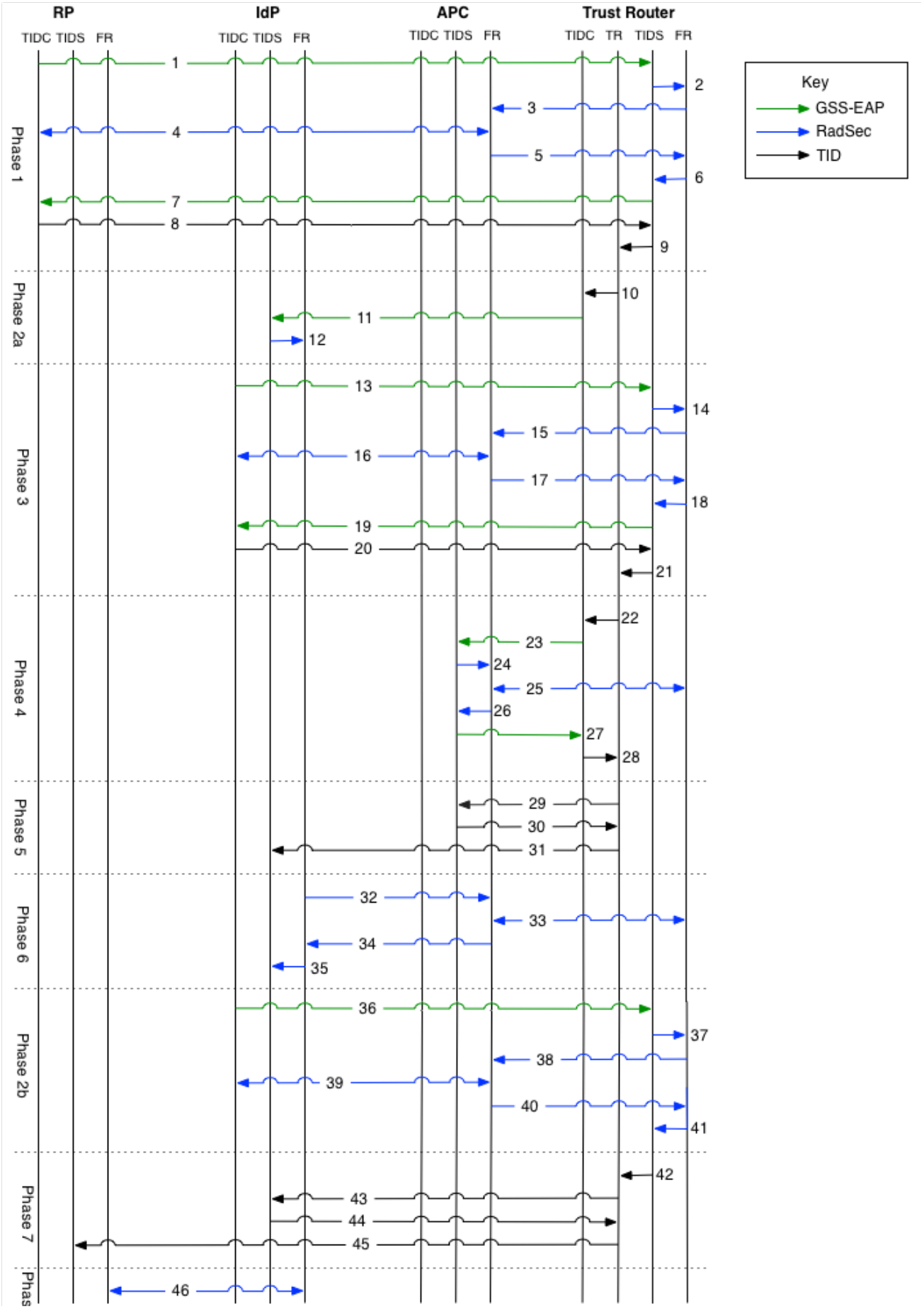
2. Detail

The steps of the protocol flow for Trust Router in the worst case (neither the RP and IdP or the IdP and APC yet share a key) are split into the following phases:

- Phase 1 - The Moonshot RP asks the Trust Router to establish keys with the Moonshot IdP that is responsible for the realm "@idp"
- Phase 2 - Mutual Authentication between Trust Router and IdP
    - Phase 2a - Trust Router authenticates to IdP
        - Phase 3 - IdP asks Trust Router to establish keys with the APC
        - Phase 4 - Mutual Authentication between Trust Router and APC
        - Phase 5 - Key exchange between IdP and APC, mediated by Trust Router
        - Phase 6 - IdP and APC connect
    - Phase 2b - IdP authenticates to Trust Router.
- Phase 7 - Key Exchange between RP and IdP, mediated by Trust Router.
- Phase 8 - RP and IdP connect

Step by step detail for each of these phases is detailed below.

If you don't know what "TIDC", "TIDS", and other such acronyms used below, then a detailed description of each of the components of Trust Router is available which should explain it all.


RP
IdP
APC
Trust Router
TIDC TIDS FR
TIDC TIDS FR
TIDC TIDS FR
TIDC TR TIDS FR
Key
GSS-EAP
RadSec
TID
Phase 1
Phase 2a
Phase 3
Phase 4
Phase 5
Phase 6
Phase 2b
Phase 7
Phas
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46

Preamble:

A Moonshot client has attempted to connect to a Moonshot enabled-service on a RP. To enable the authentication to happen, the RP must facilitate the establishment of a secure tunnel from the Moonshot Client to the relevant IdP, via its RP Proxy (go read the Moonshot introductory documentation if you're unsure how Moonshot in general works).
For the RP Proxy and IdP to communicate, they need to know each other's IP addresses and have a shared key for encrypting traffic to each other. Where this information is pre-configured in both components, great. But pre-configuring this for all instances of RPs and IdPs does not scale well. So, enter Trust Router. Trust Router can assist RPs and IdPs to securely negotiate this information in a just-in-time manner.

Phase 1 - The Moonshot RP asks the Trust Router to establish keys with the Moonshot IdP that is responsible for the realm "@idp"
Firstly, the RP asks its Trust Router to help it establish keys with the IdP responsible for the realm specified by the user attempting to access the Moonshot-enabled service. We'll call this "@idp".

- 1. The TR client on the RP wants to send a Trust Path request to the Trust Router (more specifically, to the TIDS part of Trust Router) asking for @idp. But first, it needs to authenticate to the Trust Router. To do so, it becomes a moonshot client, and as such, it establishes a GSS-EAP session to the moonshot enabled-service - TIDS on the Trust Router.
- 2. TIDS on the Trust Router, as any normal moonshot-enabled service would do, next connects to its configured RP Proxy, over RadSec, saying that the IdP in question is @apc (since the IdP for this community of Moonshot services is the APC). Here, the RP Proxy is co-located on the same server as the Trust Router.
- 3. The RP Proxy knows how to contact the APC (the small set of APCs will be pre-configured by the Trust Router adminstrator), so it forwards the request for @apc, over RadSec, to FreeRadius on the APC. Since the APC *is* @apc, it doesn't need to send the request on any further, and a TLS tunnel is established between the Trust Router client (i.e. the moonshot client) on the RP and the @apc IdP, through the Trust Router's FreeRADIUS server.
- 4. The TR client on the RP authenticates to the @apc IdP through the TLS tunnel (using the credential issued to it by the APC).
- 5. The @apc IdP sends a RADIUS Access Accept to the RP Proxy on Trust Router.
- 6. The RP Proxy forwards the Access Accept to the TIDS on Trust Router that initiated this process.
- 7. Having successfully authenticated, TIDS on the Trust Router now gives an application session to Trust Router client on the RP.
- 8. The Trust Router client on the RP now uses this session to issue a Trust Path request for @rp. This also includes the first half of a Diffie Helman (DH) key exchange.
- 9. TIDS on the Trust Router forwards this to the Trust Router process. At this point, the Trust Router will add constraints to the Temporary Identity (TID) message and apply any necessary filters on it.

Phase 2 - Mutual Authentication between TR and IdP
The Trust Router now needs to connect to the TIDS on the IdP in order to perform the next step of the DH key exchange. Of course, TIDS on the IdP won't accept connections from anyone, so the TR and the IdP must now mutually authenticate.

Phase 2a - TR authenticates itself to IdP
First, the Trust Router authenticates itself via the APC.

- 10. The Trust Router wants to authenticate to TIDS on the IdP, so it asks its Trust Router component to connect to TIDS on the IdP.
- 11. The TR client is a moonshot client and establishes a GSS-EAP session between itself and the moonshot enabled-service - TIDS on the IdP, telling TIDS that the IdP that will authenticate it is @apc.
- 12. TIDS on the IdP passes it to its RP Proxy over RadSec, co-located on the same box, telling it that the IdP is @apc. The RP Proxy on the IdP next wants establish a TLS tunnel between the client (TIDC on the Trust Router) and @apc IdP, but it does't know how to get to @apc IdP yet. So, enter trust router, again!

Phase 3 - IdP asks Trust Router to establish keys with the APC
To enable the authentication to happen, the IdP's RP Proxy must facilitate the establishment of a secure tunnel from the Moonshot Client to the relevant IdP, via its RP Proxy.

- 13. The Trust Router client on the IdP wants to send a Trust Path request to the Trust Router (more specifically, the TIDS part of Trust Router) asking for @apc. But first, it needs to authenticate to the Trust Router. So it becomes a moonshot client, and establishes a GSS-EAP session to the moonshot enabled-service - TIDS on the Trust Router.
- 14. TIDS on the TR, as any normal moonshot-enabled service would do, next connects to its configured RP Proxy, over RadSec, saying that the IdP in question is @apc. Here, the RP Proxy is co-located on the same server as the Trust Router.
- 15. The RP Proxy knows how to contact the APC (the small set of APCs will be pre-configured by the Trust Router adminstrator) , so it forwards the request for @apc, over RadSec, to FreeRadius on the APC. Since the APC *is* @apc, it doesn't need to send the request on any further, and a TLS tunnel is established between the Trust Router client on the IdP and the @apc IdP, through the Trust Router's FreeRADIUS server.
- 16. The Trust Router client on the IdP authenticates to the @apc IdP through the TLS tunnel (using the credential issued to it by the APC).
- 17. The @apc IdP sends a RADIUS Access Accept to the RP Proxy on the Trust Router.
- 18. The RP Proxy forwards the Access Accept to the TIDS on Trust Router that initiated this process.
- 19. Having successfully authenticated, TIDS on the Trust Router now gives an application session to Trust Router client on the IdP.
- 20. The Trust Router client on the IdP now uses this session to issue a TP request for @apc. This also includes the first half of a DH key exchange.
- 21. TIDS on the Trust Router forwards this to the Trust Router process. At this point, the Trust Router will add constraints to the TID message and apply any necessary filters on it.

Phase 4 - Mutual Authentication between the Trust Router and APC

The Trust Router now needs to connect to the TIDS on the APC in order to perform the next step of the DH key exchange. Of course, TIDS on the APC won't accept connections from anyone, so the Trust Router and the APC must now mutually authenticate.

- 22. The Trust Router wants to authenticate to TIDS on the APC, so it asks its Trust Router process to connect to TIDS on the APC.
- 23. The Trust Router client is a moonshot client and establishes a GSS-EAP session between itself and the moonshot enabled-service - TIDS on the APC, telling TIDS that the IdP that will authenticate it is @apc.
- 24. TIDS on the APC passes it to its RP Proxy over RadSec, co-loed on the same box, telling it that the IdP is @apc. The RP Proxy recognises that it *is* the @apc IdP, so it doesn't need to send the request anywhere else, and so a TLS tunnel is established between the client (the Trust Router process on the Trust Router) and the @apc IdP.
- 25. The TR client on the TR sends its credentials to the @apc IdP through the TLS tunnel (using the credential issued to it by the APC).
- 26. The @apc IdP sends a RADIUS Access Accept to TIDS on the same server.
- 27. Having successfully authenticated, TIDS on the APC now gives an application session to the Trust Router client on the Trust Router.
- 28. This session is passed onto the Trust Router process. At this point, the Trust Router will add constraints to the TID message and apply any necessary filters on it.

Phase 5 - Key exchange between IdP and APC

Now that the IdP and APC have mutually authenticated, the Trust Router can help negotiate a DH key exchange between the IdP and APC.

- 29. The Trust Router gives the first half of the DH key exchange that it received in step 20 to TIDS on APC
- 30. TIDS on APC responds to the Trust Router with the next step of the DH exchange.
- 31. The Trust Router passes this on to TIDS on the IdP.

Phase 6 - IdP and APC Connect, TR authenticates to IdP

Now that the IdP and APC have a shared key thanks to the magic of DH, then can now connect and the Trust Router can authenticate to the IdP.

- 32. The RP Proxy on the IdP and the APC now have a shared key, so the RP Proxy on the IdP connects to the APC over RadSec and a TLS tunnel is established between the client (the Trust Router process on Trust Router) and the @apc IdP.
- 33. The Trust Router client on the RP sends its credentials to the @apc IdP through the TLS tunnel.
- 34. The @apc IdP sends ab Access Accept to FreeRADIUS on the IdP over RadSec.
- 35. FreeRADIUS on the IdP sends an Access Accept to TIDS over RadSec.

Phase 2b - IdP authenticates to TR

Now that the TR has authenticated to the IdP, the IdP in turn authenticates itself to the TR.

- 36. The TR client on the IdP is a moonshot client and establishes a GSS-EAP session between itself and the moonshot enabled-service - TIDS on the APC, telling TIDS that the IdP that will authenticate it is @apc.
- 37. TIDS on the Trust Router, as a normal moonshot-enabled service would do, next connects to its configured RP Proxy, over RadSec, saying that the IdP in question is @apc. Here, the RP Proxy is co-located on the same server as the Trust Router.
- 38. The RP Proxy knows how to contact the APC (the small set of APCs will be pre-configured by the Trust Router administrator), so it forwards the request for @apc, over RadSec, to FreeRadius on the APC. Since the APC *is* @apc, it doesn't need to send the request on any further, and a TLS tunnel is established between the Trust Router client on the IdP and the @apc IdP, through the Trust Router's FreeRADIUS server.
- 39. The Trust Router client on the IdP authenticates to the @apc IdP through the TLS tunnel (using the credential issued to it by the APC).
- 40. The @apc IdP sends a RADIUS Access Accept to the RP Proxy on the Trust Router.
- 41. The RP Proxy forwards the Access Accept to the TIDS on the Trust Router that initiated this process.
- 42. TIDS on the Trust Router forwards this to the Trust Router process. At this point, the Trust Router will add constraints to the TID message and apply any necessary filters on it.

Phase 7 - Key exchange between RP and IdP

The RP and IdP are now able to establish keys with each other by using the Trust Router to mediate the DH process.

- 43. The Trust Router gives the first half ot he DH key exchange that it received way back in step 8 to TIDS on the IdP
- 44. TIDS on the IdP responds to the Trsut Router with the next step of the DH exchange.
- 45. The Trust Router passes this on to TIDS on the RP.

Phase 8 - RP and IdP communicate

Now that the RP and the IdP have a shared set of keys, they can communicate. The job of Trust Router is done!

- 46. The RP and IdP can now set up a direct RadSec connection, enabling the Moonshot client that initiated the whole process to authenticate through a TLS tunnel to its IdP.