

RADIUS-DNSSEC

An efficient and secure infrastructure facility for inter-domain roaming in 802.1X enabled access networks

Arjan Peddemors, Henk Eertink, Roy Arends,
Remco Poortinga, and Klaas Wierenga

Januari 2005



Telematica
Instituut

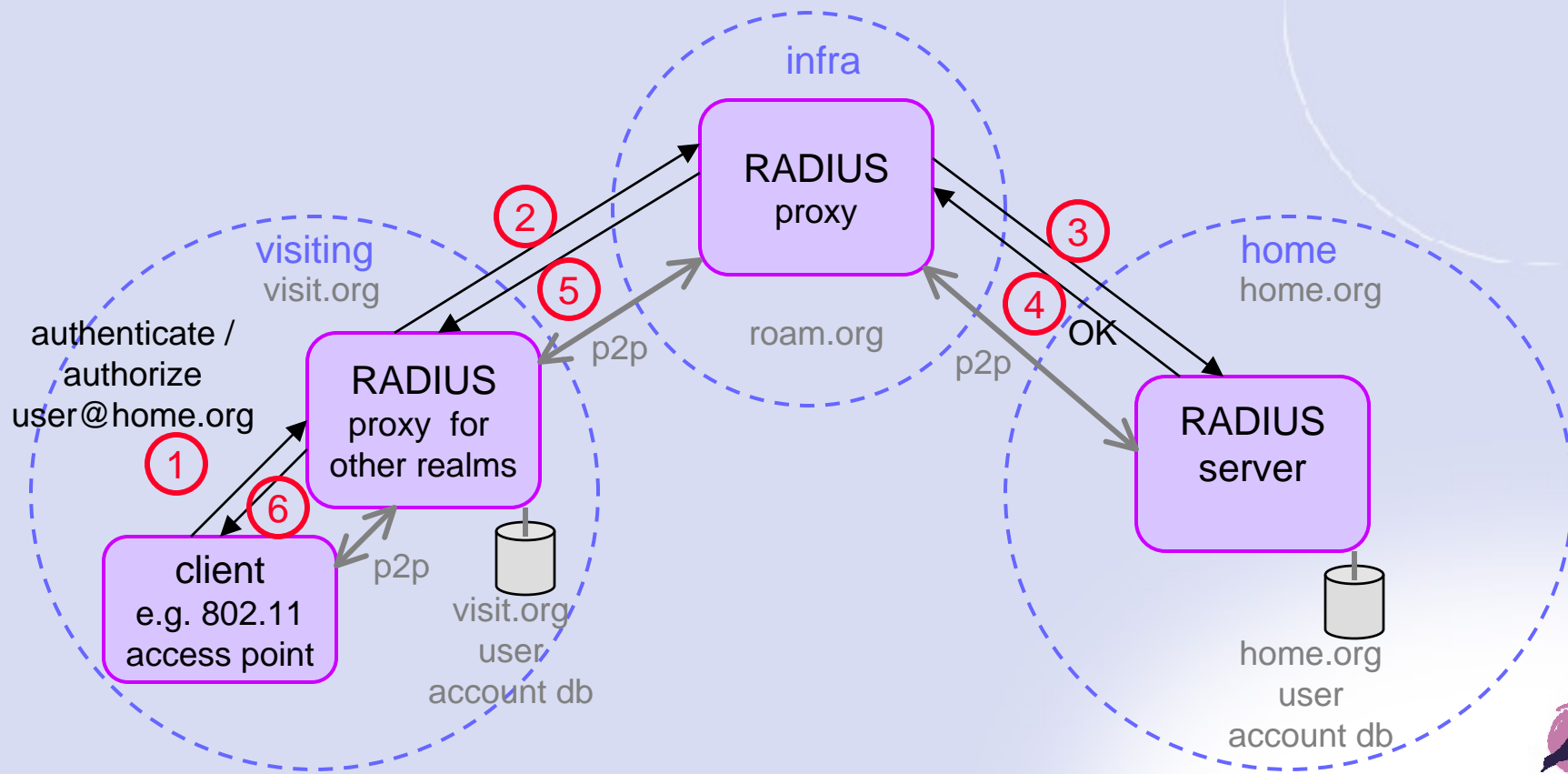
Outline

- Problem Statement (including current AA configuration)
- Overview of RADIUS-DNSSEC Solution
- Alternative Solutions
- RADIUS-DNSSEC Signaling
- RADIUS Key Exchange Protocol (RKE)
- Implementation Aspects
- Deployment Aspects



Problem Statement – Current AA Config

- Authentication and Authorization now executed through a chain of distributed RADIUS servers

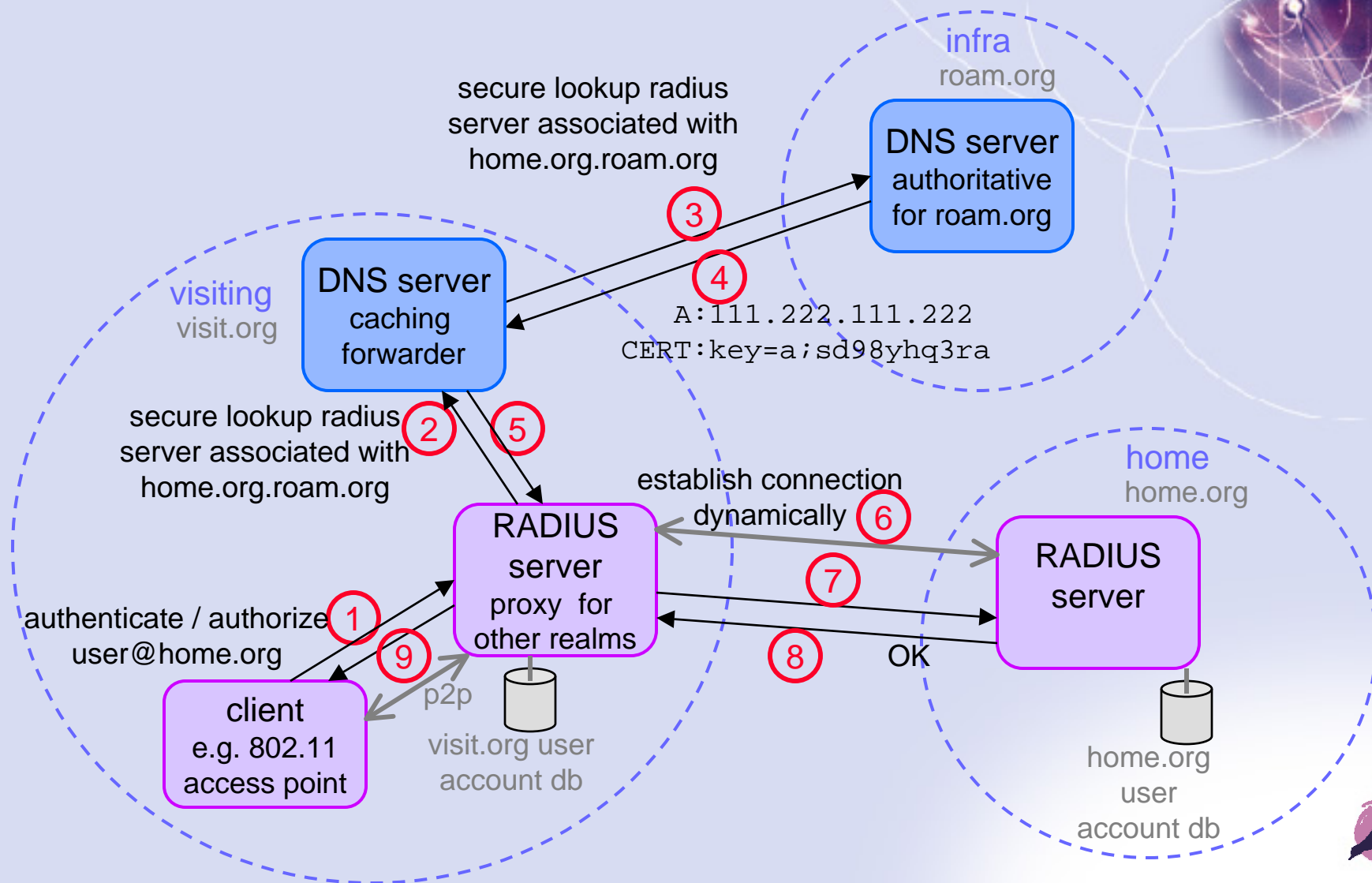


Problem Statement – Current AA Config

- RADIUS server of roaming domain administrator is linking pin between all parties involved in roaming domain
- Problems
 - Authentication traffic must travel through a chain of RADIUS proxies, while the authentication itself is only of interest to the RADIUS entities at the edges of the chain
 - Intermediate proxies may inspect the RADIUS payload which places extra requirements on the type of authentication
 - A fixed chain of proxies is error prone, as failure of one of the servers in the chain can easily result in denial of of service to roaming users
 - A shared secret must be agreed upon and exchanged out-of-band for secure communication between RADIUS peers
 - It is not easy for participating entities in a roaming agreement to obtain an overview of all other partners in the agreement



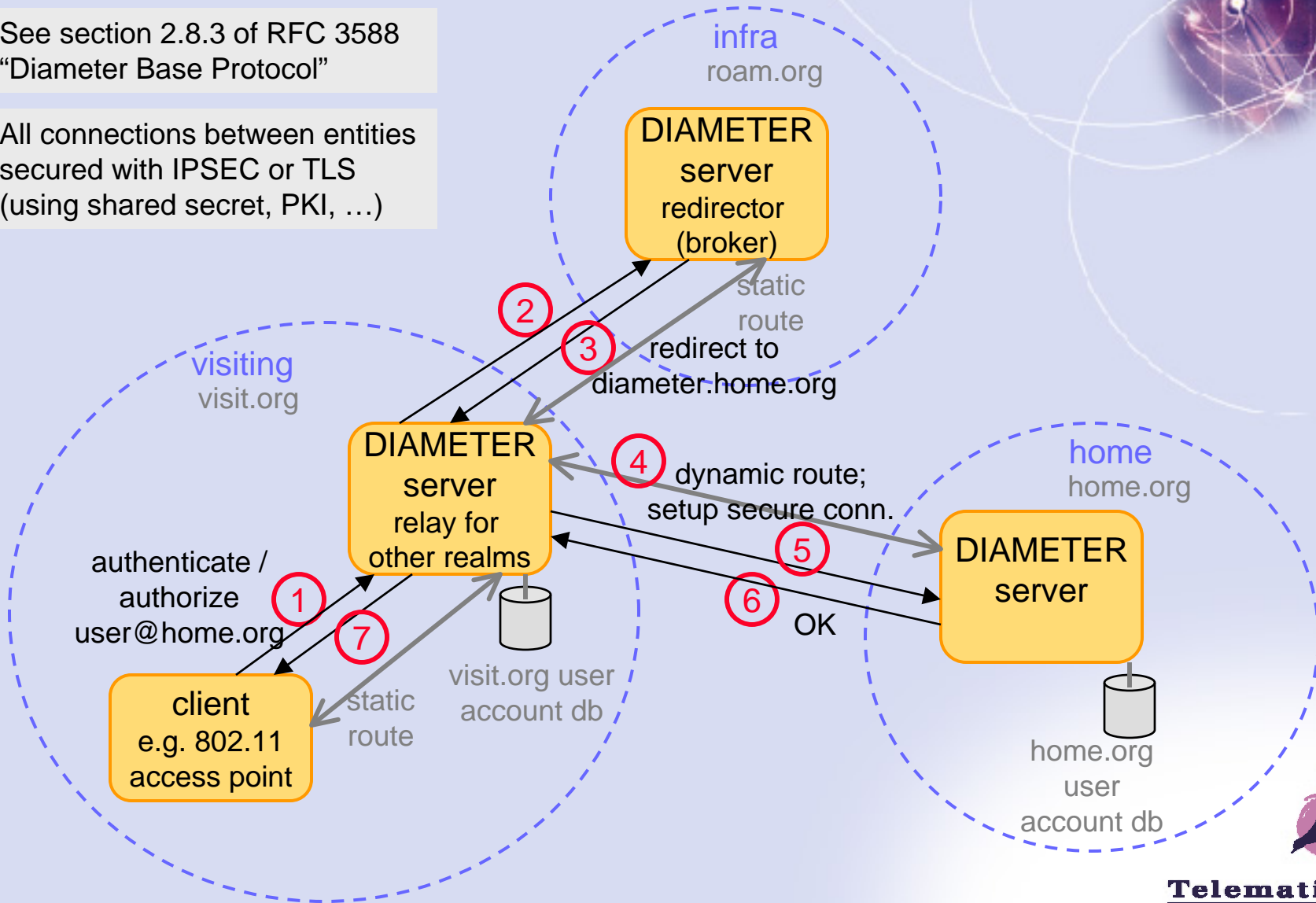
Overview of RADIUS-DNSSEC Solution



Alternative Solutions - DIAMETER

See section 2.8.3 of RFC 3588
"Diameter Base Protocol"

All connections between entities
secured with IPSEC or TLS
(using shared secret, PKI, ...)



Alternative Solutions - DIAMETER

- Benefits

- All AA traffic for roaming scenario falls within the DIAMETER protocol definition (explicit definition of broker entity)

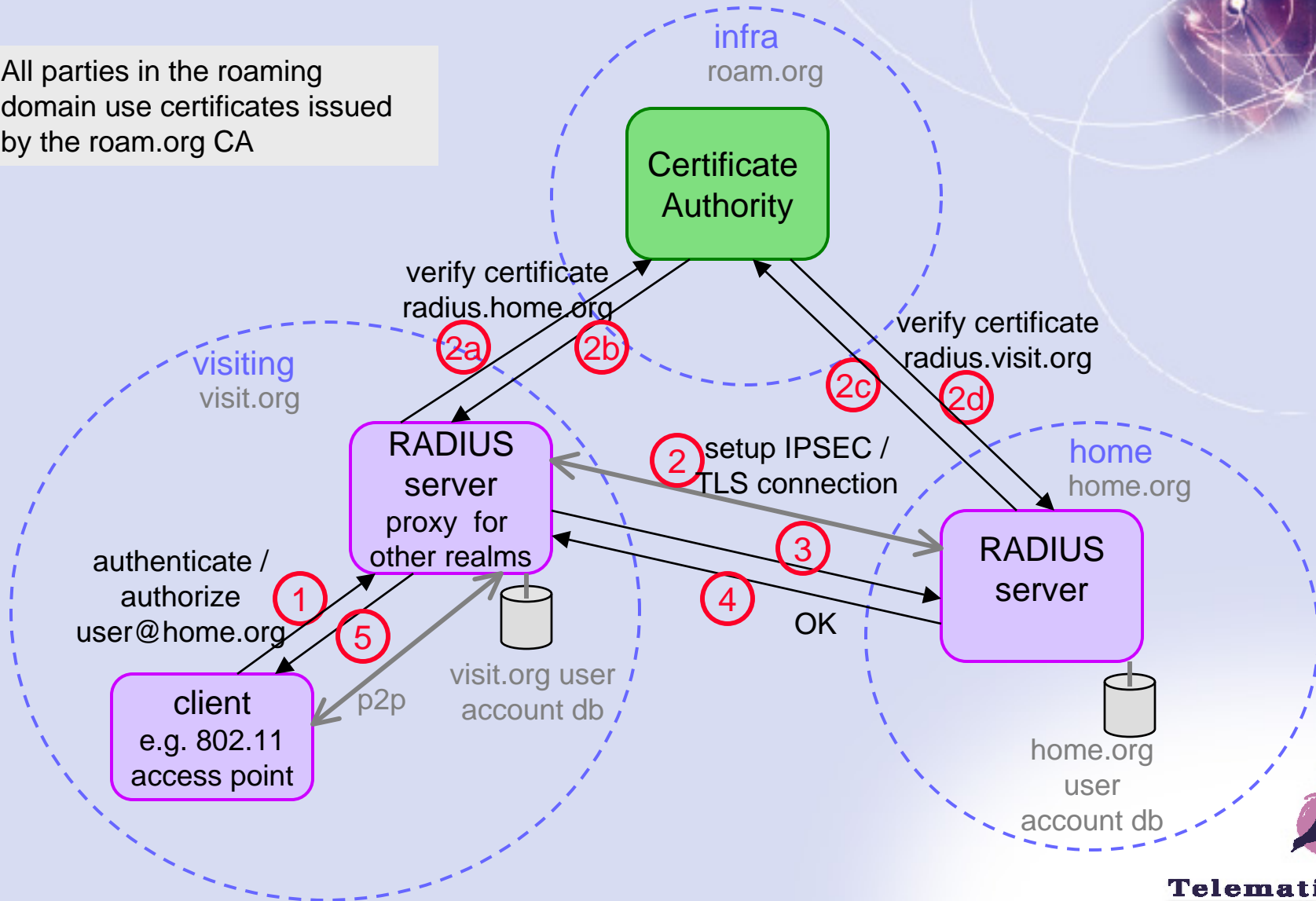
- Open Issues

- Dynamic routes established between DIAMETER entities in roaming domain most likely are secured using PKI; what about performance compared to RADIUS-DNSSEC?
- Is there a way to deny incoming AA requests from parties that are not part of the roaming domain? I.e. can the home server check that the incoming request comes from an external server that is part of the routing domain?
- Quality of implementations uncertain
- Limited deployment experience



Alternative Solutions – RADIUS / PKI

All parties in the roaming domain use certificates issued by the roam.org CA



Alternative Solutions – RADIUS / PKI

- Peer discovery details
 - Between step 1 and 2: RADIUS server of other realm (e.g. home.org) found through DNS SRV records: e.g. `_radius._tcp.home.org` or `_radius._udp.home.org` (note: RADIUS protocol defines UDP traffic only)
 - Possible alternative approach:
 - Lookup certificate at CA with realm name as key (possible?)
 - If exists, realm is part of roaming domain
 - Use additional info in certificate to determine RADIUS server of realm
 - Interaction becomes slightly different (no need for client side certificate lookup during TLS connection establishment; already done)
- Open Issues
 - What about taking part in multiple roaming domains? RADIUS server has multiple certificates; which one to provide during TLS connection establishment?
 - No implementations; custom-made solution



RADIUS-DNSSEC signaling

- Three distinct RADIUS entities: client, proxy, (authenticating) server
- RADIUS client uses local proxy RADIUS server for authentication of user (client-server authentication between these RADIUS entities based on shared secret)
- No specific requirements for type of authentication
- User authentication is *realm* based where the user name contains a portion that identifies the user's home domain, e.g. jan@**home.org**
- The realm identifier in the user name typically refers to a DNS domain name
- Local users, i.e. users that are part of the local realm, are authenticated and authorized by the local proxy RADIUS server
- Non-local users, i.e. users not part of the local realm, must be authenticated and authorized by the RADIUS server in the user's home domain

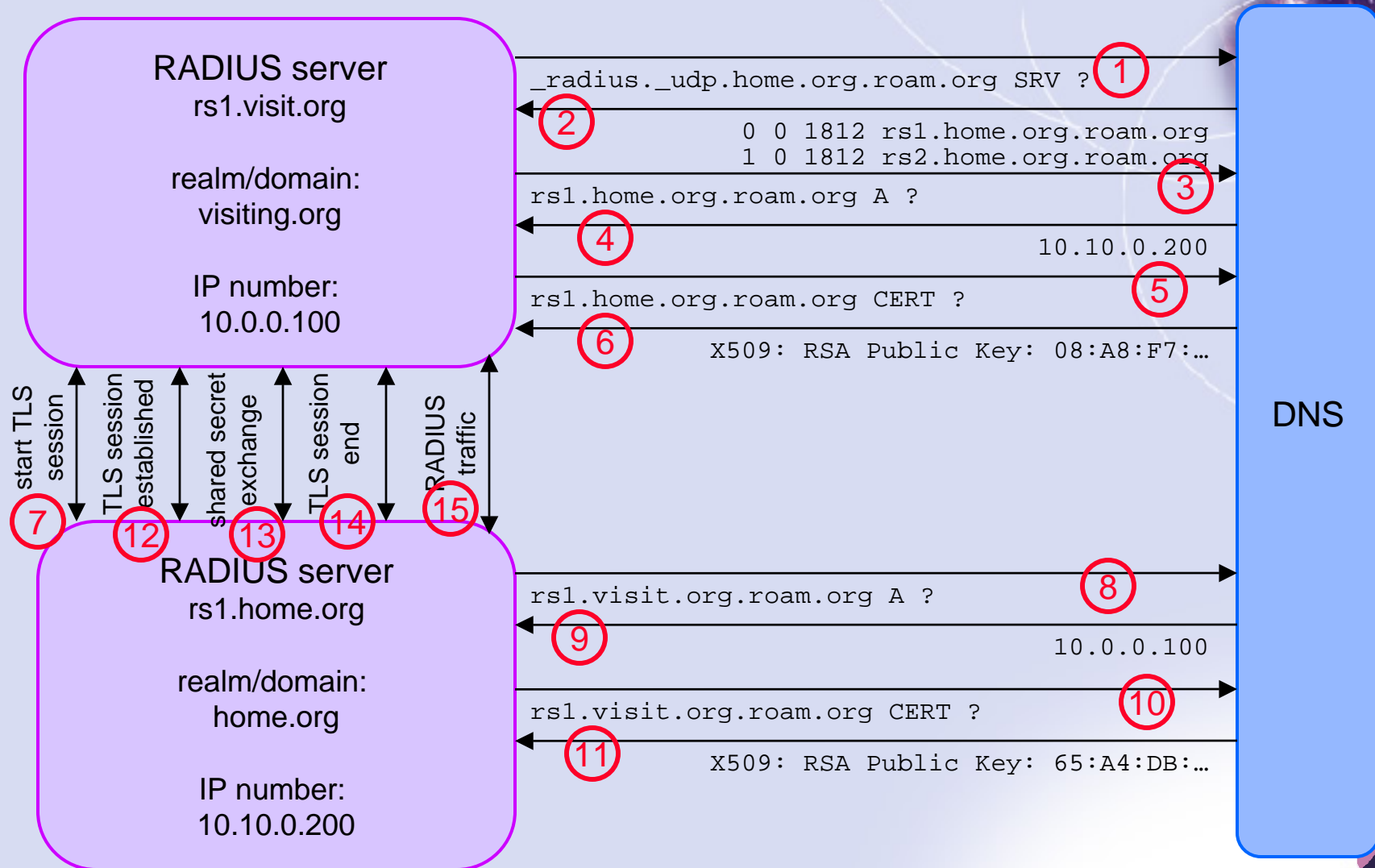


RADIUS-DNSSEC signaling

- For non-local users, the proxy server obtains the information on the authenticating server by looking up the realm and associated server details through DNSSEC in the domain of the roaming broker
- Usage of DNSSEC ensures DNS data authenticity and integrity; spoofing could lead to access to the roaming domain by unauthorized parties
- Each individual domain that takes part in the roaming domain has one or more RADIUS servers that authenticate and authorize the users from that domain (multiple servers for redundancy)
- Every RADIUS server has its own public and private key pair and has its public key published in the form of a certificate in the DNS in the roaming domain DNS tree
- The RADIUS proxy needs to setup a dynamic secure connection with the authenticating RADIUS server: this is accomplished by first exchanging a shared secret and then use this secret to have a regular RADIUS conversation



RADIUS-DNSSEC signaling



RADIUS-DNSSEC signaling

- 1) The `visit.org` proxy RADIUS server needs to authenticate a user from `home.org` and therefore needs to know an authenticating RADIUS server for the `home.org` domain. The DNS server of `roam.org` holds this information; the proxy sends a SRV DNS query for `_radius._udp.home.org.roam.org` (concatenating `_service`, `_protocol`, `realm`, and `roaming domain`). Using SRV RRs at this points allows for additional authentication services in the future, such as Diameter. Note that the proxy may be part of one or more roaming domains and from the credentials supplied by the user it is not clear which roaming domain is most appropriate. Therefore, we assume that the RADIUS entities taking part in multiple roaming agreements employ a roaming domain list in decreasing order of preference, where an authenticating RADIUS server for the user's home domain is tried to be found through the roaming domain DNS server with the highest priority first, then, if not successful, the second highest priority roaming domain DNS server, etc.
- 2) DNS returns a list of SRV RRs describing potential authenticating RADIUS servers for `home.org`. Each entry is associated with a priority value; the proxy selects the server with the lowest priority value to setup a connection. If this connection does not succeed later, the proxy selects the server with the second lowest priority, etc. If no server is available or all servers are disqualified due to an error, this procedure stops with an error. In this example, it selects the `rs1.home.org.roam.org` server with default RADIUS port 1812. If a shared secret is already available for the selected server and the time-to-live has not expired, go to step 15 immediately.
- 3) Lookup the IP address of the selected authenticating RADIUS server (A RR).



RADIUS-DNSSEC signaling

- 5) Lookup the X.509 certificate holding the public key of the selected authenticating RADIUS server (CERT RR). The X.509 certificates for the RADIUS servers stored under the `roam.org` tree all are version 3 or higher and use the *Subject Alternative Name* certificate extension (see RFC2459, section 4.2.1.7). The alternative name of each RADIUS server certificate is of type `dNSName` and indicates the server's DNS entry in the roaming domain tree (here, `rs1.home.org.roam.org`). This makes the certificate self referring through DNS: when the certificate is known, it is possible to obtain a copy using the alternative name. (Investigate; maybe it is possible to use a DNS name in the regular X.509 certificate Subject Name). The server certificates do not need to be signed by a particular party and may even be self signed. The fact that they are stored in the `roam.org` DNS tree provides trust to all participants in the roaming domain that the certificates are correct.

- 7) Now the proxy has all information needed to contact and authenticate the selected authenticating server in the `home.org` domain, but for a regular RADIUS conversation to take place (to authenticate the user) it needs to exchange a shared secret. This is executed using the Radius Key Exchange (RKE) protocol, running on top of TLS (see slides further on). Step 7 indicates the start of the TLS session. In addition to listening on its UDP port for incoming RADIUS requests, the authenticating RADIUS server accepts TCP connections on port 5436 for RKE interaction. The proxy connects to TCP port 5436 of `rs1.home.org`, and starts the TLS session.



RADIUS-DNSSEC signaling

- 7) (continued) Concurrent RKE sessions may take place to the same peer, for instance, when two users from different domains are cross visiting each others domains. This could lead to out of sync shared secrets between the RADIUS peers: see also the description of step 15 for handling this case. The TLS handshake mechanism takes care of exchanging certificates. However, in TLS the server (authenticating RADIUS server) must send its certificate before the client (proxy RADIUS server), but the server does not know anything about the client yet. The server may have different certificates for different roaming domains and therefore needs to know from which roaming context the TLS session is being initiated. Therefore, we use the *TLS Server Name Indication* extension (see RFC3546, section 3.1). The client sends the server name as found in the `roam.org` domain to the server, here `rs1.home.org.roam.org`. Now, the server can send its `roam.org` certificate. Additionally, the server indicates it wishes to receive a client certificate. The client receives the server certificate and checks that it matches with the one retrieved from `roam.org`. It then sends back its own certificate and sets the *Certificate Verify* message (for client authentication).
- 8) The authenticating server interprets the RADIUS proxy certificate. It knows from the alternative name the client's DNS name in `roam.org`: `rs1.visit.org.roam.org`. It queries DNS to obtain the address of the client and checks that it matches the IP address of the connection.

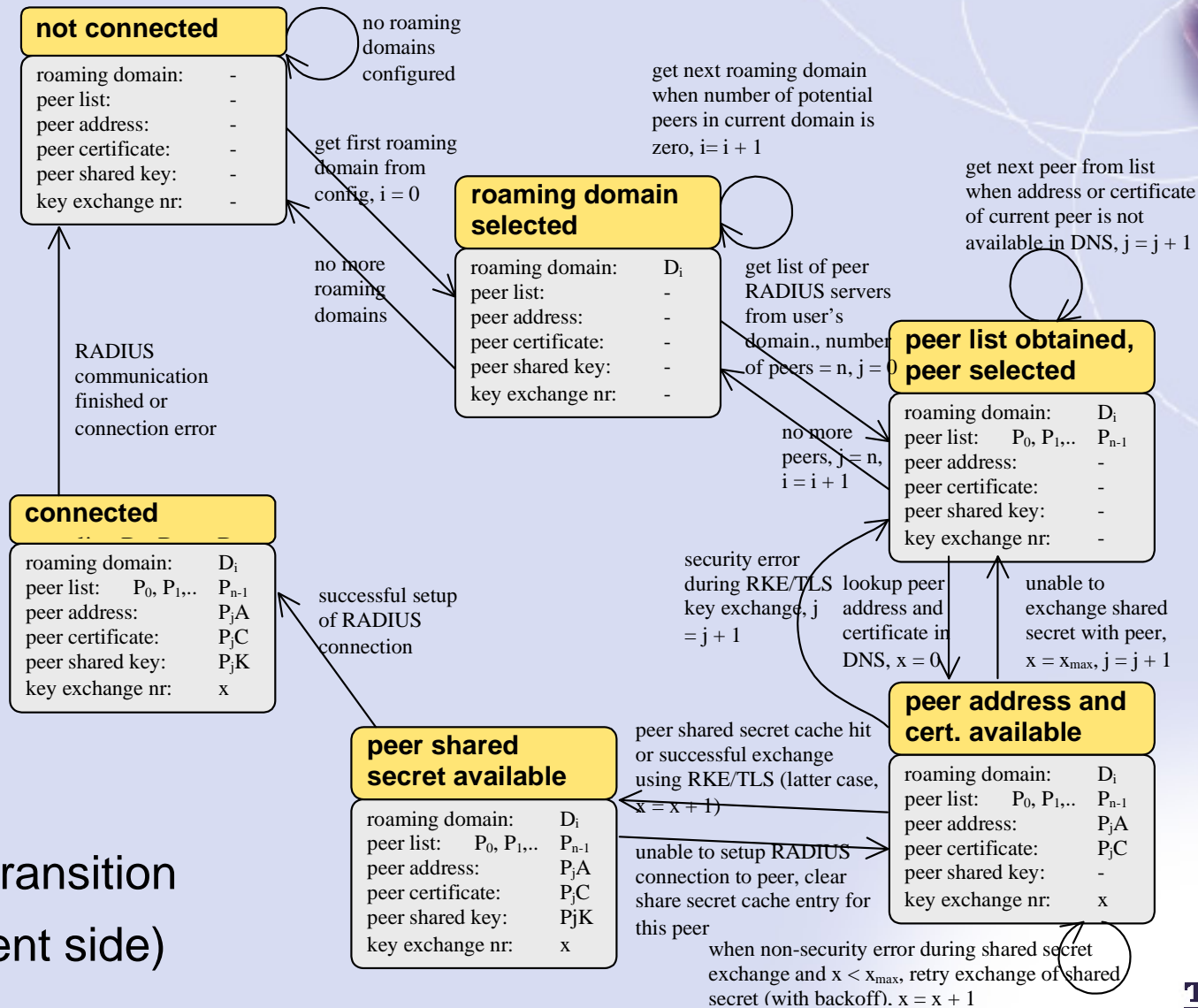


RADIUS-DNSSEC signaling

- 10) The authenticating server obtains the certificate of the proxy and checks that it matches with the certificate obtained during the TLS handshake procedure.
- 12) Now all checks are finished and the TLS session is established.
- 13) Exchange the shared secret using the RKE protocol.
- 14) End the TLS session; at this point the shared secret is available at both sides for regular RADIUS communication.
- 15) Perform RADIUS communication using the regular RADIUS peer mechanism. If the peer connection between the two RADIUS servers cannot be established due to an authentication error (for instance due to out of sync keys), the proxy tries to exchange a new shared secret (going back to step 3) . However, for every new RADIUS peer session, a new shared secret will be exchanged only an x_{\max} maximum number of times; a peer authentication error after x_{\max} number of successful shared secret exchanges disqualifies the authenticating RADIUS server. After disqualification, another authenticating server will be selected (going back to step 2), which is next in order of priority. Successive key exchanges take place after a short random backoff period (say between 10ms and 100ms), to prevent concurrent key exchange loops.



RADIUS-DNSSEC signaling

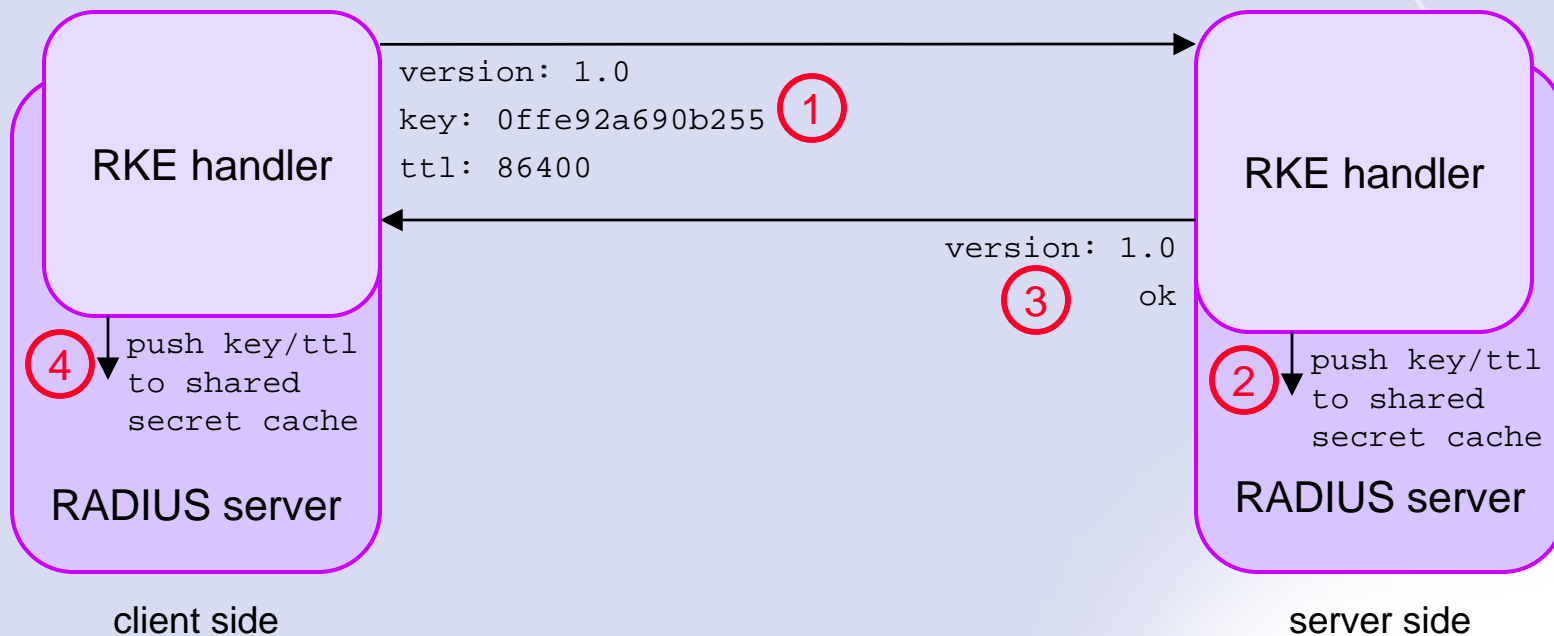


state transition
(client side)



RADIUS Key Exchange Protocol (RKE)

- Straightforward exchange of a shared secret between RADIUS peers, assuming a secure reliable channel
- No concurrent RKE sessions may take place to the same peer



RADIUS Key Exchange Protocol (RKE)

RKE interaction

- 1) After establishment of the reliable secure channel, the client sends the RKE version information, the proposed shared key and the time-to-live value to the server. The key is a random 128-bits value. This message consists of three lines of ascii tekst: 'version: 1.0' followed by CRLF, followed by 'key: ' followed by the 128-bits value in hexadecimal notation followed by CRLF followed by 'ttl: ' followed by the number of seconds for the ttl (decimal, value less than max. of unsigned 32-bit integer) followed by CRLF
- 2) The server pushes the key and ttl value to the cache of the RADIUS server
- 3) The server replies (ascii) with 'version: 1.0' followed by CRLF followed by 'ok' or 'error' followed by CRLF. In case of error, the procedure is stopped and no keys/ttl pairs are stored
- 4) The client pushes the key and ttl value to the cache of the RADIUS server

Open issues / investigate

- It may be possible to use the shared secret generated for the TLS session as RADIUS shared secret
- Due to a TLS connection interruption or other error, the server may store the shared secret but not the client; this situation is covered by the extra key exchange attempts in case the radius peers do not successfully connect



Dynamic Key Exchange - Alternatives

- RADIUS traffic over TLS
 - No need to exchange shared RADIUS secret; all traffic is over secure TLS connection
 - Public keys from DNS used for mutual authentication
 - TLS uses reliable transport (for most practical cases: TCP), but RADIUS is UDP based
 - Possible mismatch between flow control and error recovery functionality in RADIUS server implementations and similar functionality in TCP when simply replacing UDP with TCP sockets
 - Flow control and error recovery functionality probably hard to remove from RADIUS server implementation: mixed with application logic
 - Conclusion: high implementation effort



Dynamic Key Exchange - Alternatives

- RADIUS traffic over DTLS (Datagram Transport Layer Security)
 - No need to exchange shared RADIUS secret; all traffic is over secure DTLS connection
 - Public keys from DNS used for mutual authentication
 - Current (single) implementation not (yet) of high quality
 - Conclusion: no suitable implementation available
- IPSec between RADIUS servers
 - No need to exchange shared RADIUS secret; all traffic between hosts is protected by IPSec
 - Public keys from DNS used for mutual authentication
 - Requires system-level configuration on the RADIUS servers (extra deployment effort)
 - All traffic between these hosts is encrypted; may be inappropriate when additional services run on these hosts
 - Conclusion: good alternative



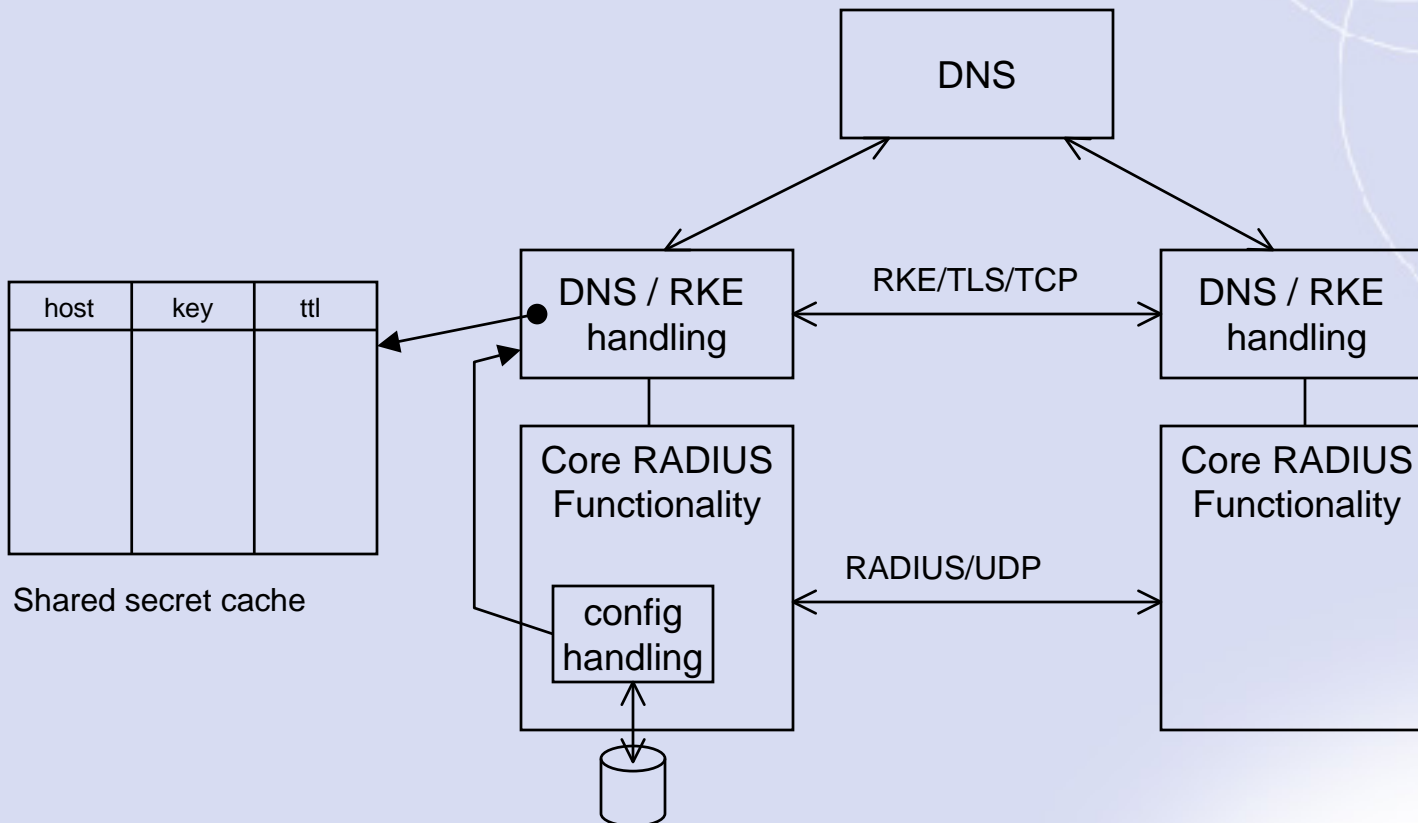
Implementation Aspects

- TBD
- Things like: placing caching forwarder DNS server on same host as RADIUS proxy, because of resolver library limitations



Implementation Aspects

- Functional Decomposition



Deployment Aspects

- TBD
- Things like: which procedures to follow to populate the roam.org domain with data
- Have to deal with DNSSEC “islands of trust”
- Running RADIUS-DNSSEC in combination with existing “chained” solution

