Freifunk DNSSEC & sig0namectl

Gentle Introduction

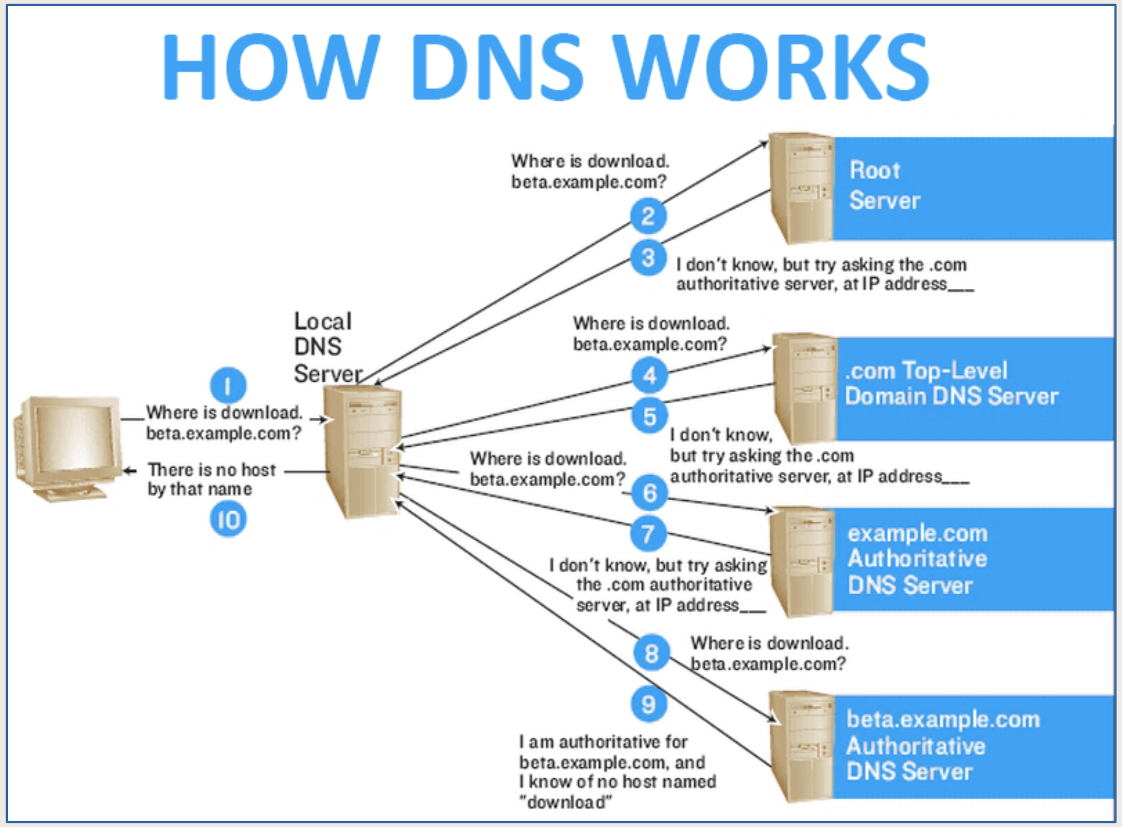
- DNS – how it works
- DNSSEC – what is it good for?
- Freifunk DNSSEC deployment – the story so far

sig0namectl

- Dynamic DNS with SIG(0) keypairs
- Wide Area DNS-SD
- Example demo

Future Plans & Open Discussion

DNS: talkin' bout a resolution …


HOW DNS WORKS
Root Server
.com Top-Level Domain DNS Server
example.com Authoritative DNS Server
beta.example.com Authoritative DNS Server
Local DNS Server
1 Where is download. beta.example.com?
2 Where is download. beta.example.com?
3 I don't know, but try asking the .com authoritative server, at IP address___
4 Where is download. beta.example.com?
5 I don't know, but try asking the .com authoritative server, at IP address___
6 Where is download. beta.example.com?
7 I don't know, but try asking the .com authoritative server, at IP address___
8 Where is download. beta.example.com?
9 I am authoritative for beta.example.com, and I know of no host named "download"
10 There is no host by that name

from https://techreader.com/files/2016/12/how-the-dns-system-works.png

DNS Server deployment types:

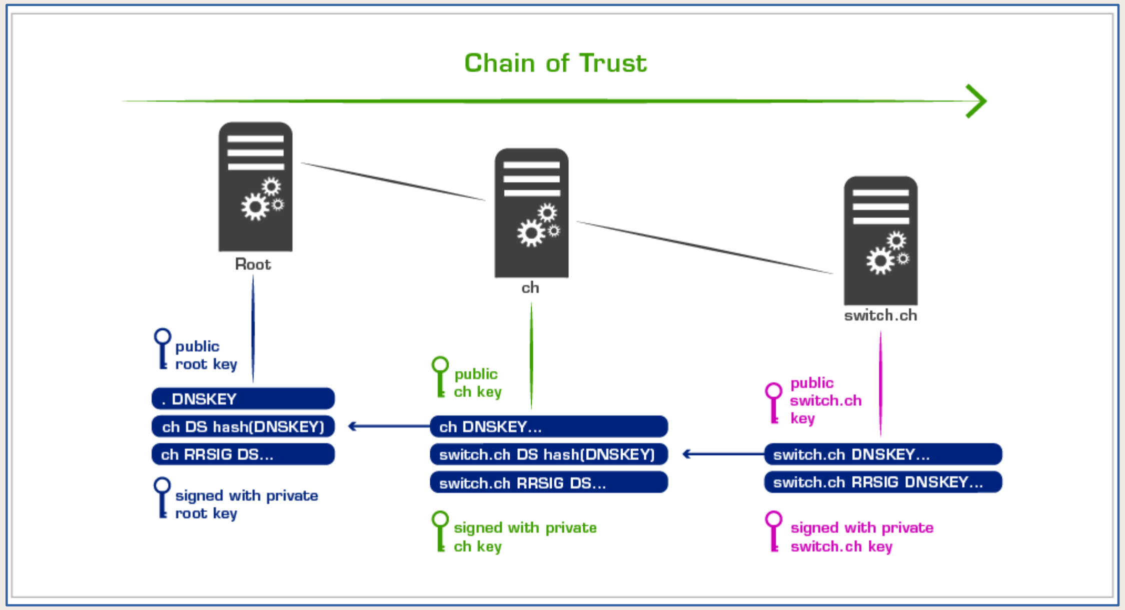
- **recursive caching resolver** servers (usually local, sometimes public, eg Google)
- **authoritative zone** servers (usually global)

DNS requests are usually of type **query** (read), but **update** (write) requests can also be issued.

- request resolution (query or update)
- get answer (result)

DNS servers respond to query & update requests for resource record types, e.g. **A records** for IPv4, **AAAA records** for IPv6 & **MX records** for mail server address resolution. There is much, much more. **`nsupdate -T`** gives the full list of DNS resource record types.

DNSSEC: DNS Anchored Chain of Trust



from https://www.nic.ch/export/shared/.content/images/dnssec_chain_en.png

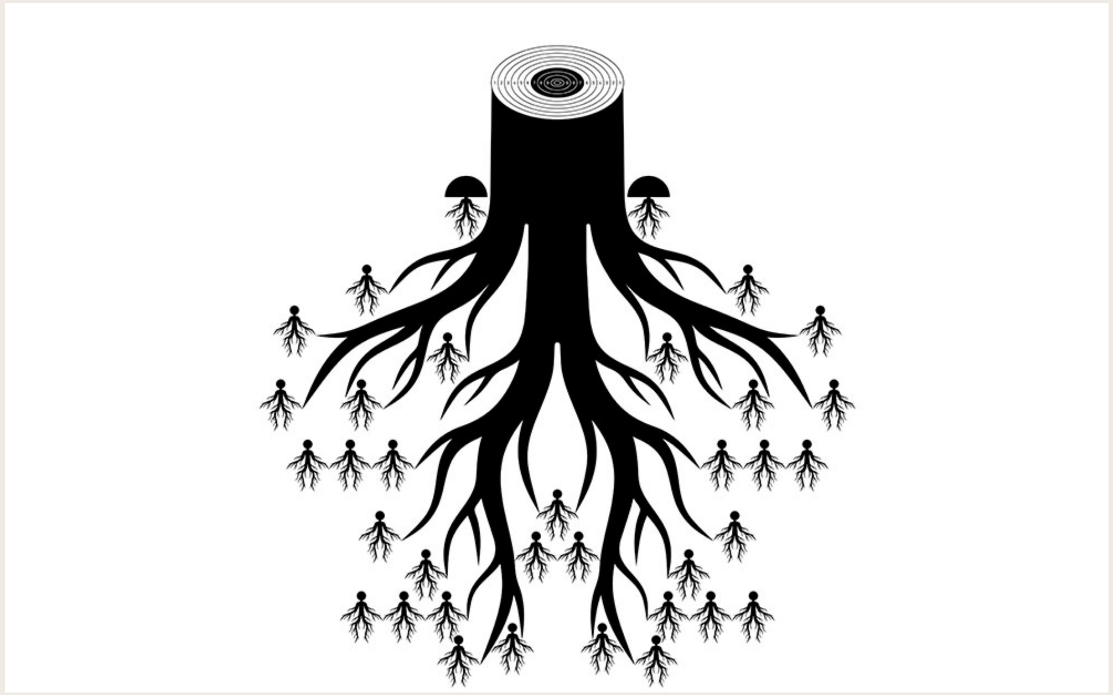
- DNSSEC introduces a cryptographic chain of trust with the **DNS root key** as the trust anchor.
- The DNS root key is securely updated in an open key signing ceremony.
- DNSSEC offers cryptographic proof of integrity for data returned by DNSSEC enabled servers.

DNSSEC: Freifunk deployment

The story so far ...

- Freifunk DNS admin Werner recently retired so plans were made to migrate the primary DNS server and hand over admin to nosy.
- Migration is done, lets hear from nosy ...

sig0namectl: to name is to own



MYCOsystem by Małgorzata Gurowska from https://culture.pl/en/artist/malgorzata-gurowska
(permission pending from artist to use as sig0namectl logo)

sig0namectl is a project designed to allow community networks to directly & securely publish and update their own information and services into the community's DNS system.

The project contains utilities to ease publishing and updating workflows that can be run on personal laptops, Freifunk routers and even in web browsers.

It leverages DNSSEC, SIG(0) DNS keys, wide area DNS Service Discovery and much more to allow participants to update all their public resource records in a reliable, scalable and open standards-based method.

sig0namectl: Project Motivations

Autonomy

- minimise external dependencies.
- minimise extractive proprietary external APIs
- offers an initial trust root through which others can grow.

Sustainability

- minimal deployment footprint
  (DNS is already around us & flows like water)
- scale use as you need and your community grows

Freedom

- opportunity for local & community network to offer services
- alternatives to using centralised data centre infrastructures
- remove unnecessary extractive 3rd party services

SIG(0) DNS KEY Authentication

```
$ dig +noall +answer +multiline vortex.zenr.io KEY
vortex.zenr.io.         600 IN KEY 512 3 15 (
         2MK3KZkUgYQVumU9bhy1KzIZ2FhFQZ8yLP2nFMJRCEQ=
                         ); alg = ED25519 ; key id = 56161
```

Using **dig** to get further information about public KEY resource records

The first step in using sig0namectl is to generate a named SIG(0) keypair and to request registration under a compatible domain.

SIG(0) is a modern standard authentication standard that uses a public and private key pair to authenticate and sign DNS update requests.

The public key is published as a DNS **KEY** resource record.

The private key should kept secure in the owner’s local host that generated the key pair.

New KEY Generation & Request (client)

```
$ ./request_key zembla.beta.freifunk.net
Generating key pair.
Kzembla.beta.freifunk.net.+015+37757
New SIG0 keypair for zembla.beta.freifunk.net generated in
./keystore
KEY request 'zembla._signal.beta.freifunk.net  IN KEY 512
3 15 dc2/whMCewe4NAUqNdBURBHEa4ykDPSgguYIUqhqOcA=' added
$
```

Using the sig0zonectl tool **request_key** to generate a new KEY resource record for zembla.beta.freifunk.net

```
$ dig +noall +answer zembla.beta.freifunk.net KEY
zembla.beta.freifunk.net. 54 IN KEY 512 3 15 (
           dc2/whMCewe4NAUqNdBURBHEa4ykDPSgguYIUqhqOcA=
           ); alg = ED25519 ; key id = 37757
```

Using **dig** to get further information about the successfully registered KEY zembla.beta.freifunk.net

Using the sig0zonectl tool **request_key** tool, a client generates a KEY pair locally with a unique label (zembla) beneath an existing compatible domain (beta.freifunk.net). The tool then sends a DNS update request to the primary DNS server responsible for the domain to add the public key resource record to the domain.

The DNS server may grant or ignore this request.

The registration request is published as a DNS **KEY** resource record in a holding name space for review. The default DNS server automatic policy is to accept new unique name requests on a “first-come, first served” basis, to accept if the fully qualified domain name (FQDN) is unused.

If successful, the new KEY is published and the default policy applied is that the keypair can be used to update records at and below the KEY’s keyname FQDN.

KEY Request Acceptance (DNS Admin)

```
$ ./process_requests zembla.beta.freifunk.net

LIST of  KEY REQUESTS
_signal.zenr.io.    PTR zembla._signal.beta.freifunk.net.

PROCESSING KEY REQUESTS
KEY 'zembla.beta.freifunk.net' added under zone
'beta.freifunk.net' with '[key id = 23799]', IDN
'zembla.zenr.io'.
```

Using **process_requests** to register a new unused KEY with FQDN zembla.zenr.io

```
$ ZONE="zenr.io" ./process_requests

LIST of  KEY REQUESTS
_signal.zenr.io.        PTR     vortex._signal.zenr.io.

PROCESSING KEY REQUESTS
KEY 'zembla.beta.freifunk.net' IS NOT added under zone
'beta.freifunk.net', as DNS resource records already
exist.
```

Using **process_requests** a second time denies the request as a resource record already exists

With access to the **zenr.io** private KEY, the zone admin can use or automate the **process_requests** tool to grant KEY requests given by clients using **request_key** on a "first come, first served" (FCFS) policy basis.

The default action is that if the KEY FQDN has no current resource records *at all*, then it is replicated to the upstream domain and SIG(0) update access is granted for the KEY FQDN and all subdomains.

If any DNS resource record already exists, then the addition request made for publishing the KEY is denied. Other policies can be configured.

sig0namectl Workflow: dynamic IP

```
$ ./dyn_ip zembla.beta.freifunk.net 10.31.40.7
zembla.beta.freifunk.net. 60    IN      A       10.31.40.7
zembla.beta.freifunk.net. 60    IN      RRSIG   A 13 4 60
20240509193324 20240429145552 60365 beta.freifunk.net.
xIJrbvrRfDLM2zcGMnu8lI+MudP75/E8l/ddL/pthU+LcY4rnLQpTVdE
lBX+WuY7hJNNQG9Zfik2m032XOqlEA==
$
```

Using the sig0zonectl tool **request_key** to generate a new KEY resource record for zembla.beta.freifunk.net

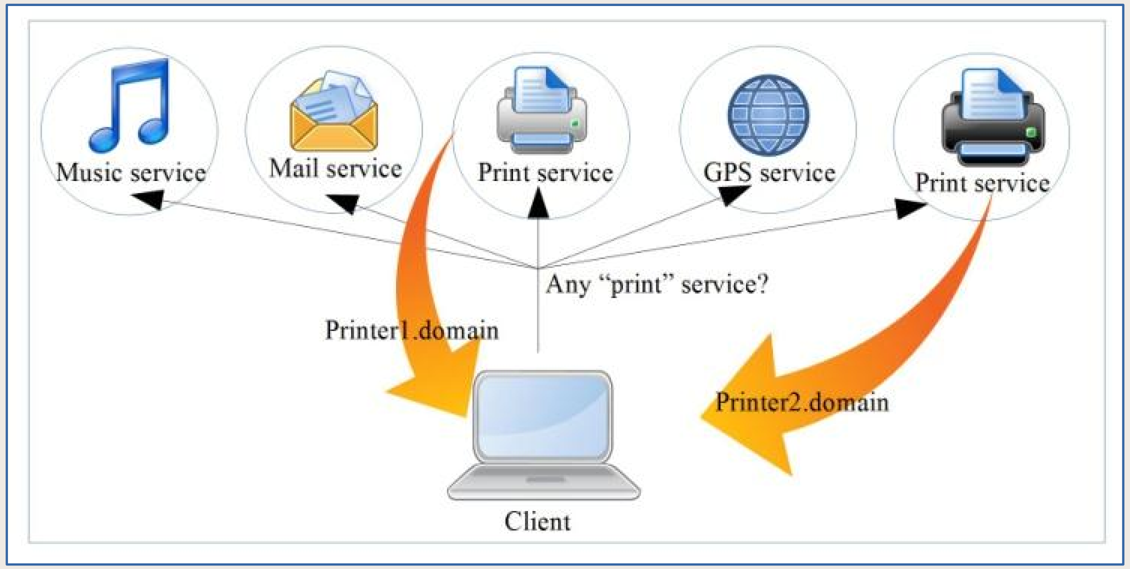
```
$ dig +noall +answer zembla.beta.freifunk.net KEY
zembla.beta.freifunk.net. 54 IN KEY 512 3 15 (
           dc2/whMCewe4NAUqNdBURBHEa4ykDPSgguYIUqhqOcA=
           ); alg = ED25519 ; key id = 37757
```

Using **dig** to get further information about the successfully registered KEY zembla.beta.freifunk.net

dyn_ip: The simplest sig0namectl tool is the usual form of dynamic IP address management featuring:

- Assignment of multiple IPv4 and IPv6 addresses
- Assign IP addresses for any FQDN below the keyname (example keyname given is zembla.beta.freifunk.net)

WA DNS-SD (Wide Area Service Discovery)



Service Browsing example for print services
from https://www.researchgate.net/publication/279177017_Internet_of_Things_A_Survey_on_Enabling_Technologies_Protocols_and_Applications

```
$ORIGIN _dns-sd._udp.zembla.beta.freifunk.net.
b           PTR         zembla.beta.freifunk.net.
db          PTR         zembla.beta.freifunk.net.
dr          PTR         zembla.beta.freifunk.net.
lb          PTR         zembla.beta.freifunk.net.
r           PTR         zembla.beta.freifunk.net.
_services   PTR         _printer._tcp.zembla.beta.freifunk.net.
_services   PTR         _http._tcp.zembla.beta.freifunk.net.
$ORIGIN _printer._tcp.domain.
            PTR         Printer1.zembla.beta.freifunk.net.
            PTR         Printer2.zembla.beta.freifunk.net.
```

DNS-SD provides for 3 core functions within a DNS domain that include the use of further DNS **PTR**, **TXT** & **SRV** resource records for:

- service registration (service offering)
- service enumeration (service & type browsing)
- service use (service resolving/lookup)

DNS-SD offers these functions over two core transport mechanisms for two distinct scopes:

- multicast DNS for local broadcast domains (for .local addresses)
- regular unicast DNS for global domains (Wide Area DNS-SD domains)

Workflow: DNS-SD Domains

```
$ ./dnssd-domain zembla.beta.freifunk.net
$
```

Using the sig0zonectl tool **dnssd-domain** to establish DNS-SD browsing & registration PTR resource records for a domaon

dnssd_domain offers configuration of DNS-SD PTR records to allow browsing & registration of services within a domain.

```
$ORIGIN _dns-sd._udp.zembla.beta.freifunk.net.
b                    PTR     zembla.beta.freifunk.net.
lb                   PTR     zembla.beta.freifunk.net.
db                   PTR     zembla.beta.freifunk.net.
r                    PTR     zembla.beta.freifunk.net.
dr                   PTR     zembla.beta.freifunk.net.
$
```

Listing of zone PTR updates under zembla.zenr.io after above command

For any particular domain, DNSSD-aware aware software query these PTR resource records underneath `_dns-sd._udp` to determine the domains to query for service browsing and service registration.

Workflow: DNS-SD service types

```
$ DNSSD_SERVICES="_http._tcp" \
./dnssd-service zembla.beta.freifunk.net
$
```

The sig0zonectl **dnssd-service** script gives examples of how to register DNS-SD service types and instances for browsing & enumeration within a domain

```
$ avahi-browse -brat -d zembla.beta.freifunk.net
...
```

The dig or avahi-browse Linux tools can be used to enumerate details of all service types and instances registered within a domain (long output not shown)

```
$ avahi-browse -tr _http._tcp  -d zembla.beta.freifunk.net
+    n/a  n/a zembla  _http._tcp  zembla.beta.freifunk.net
=    n/a  n/a zembla  _http._tcp  zembla.beta.freifunk.net
   hostname = [zembla.beta.freifunk.net]
   address = [2a01:4f8:c17:3dd5:8000::10]
   port = [80]
   txt = ["Hello_this_is_a__http._tcp_service_instance"]
```

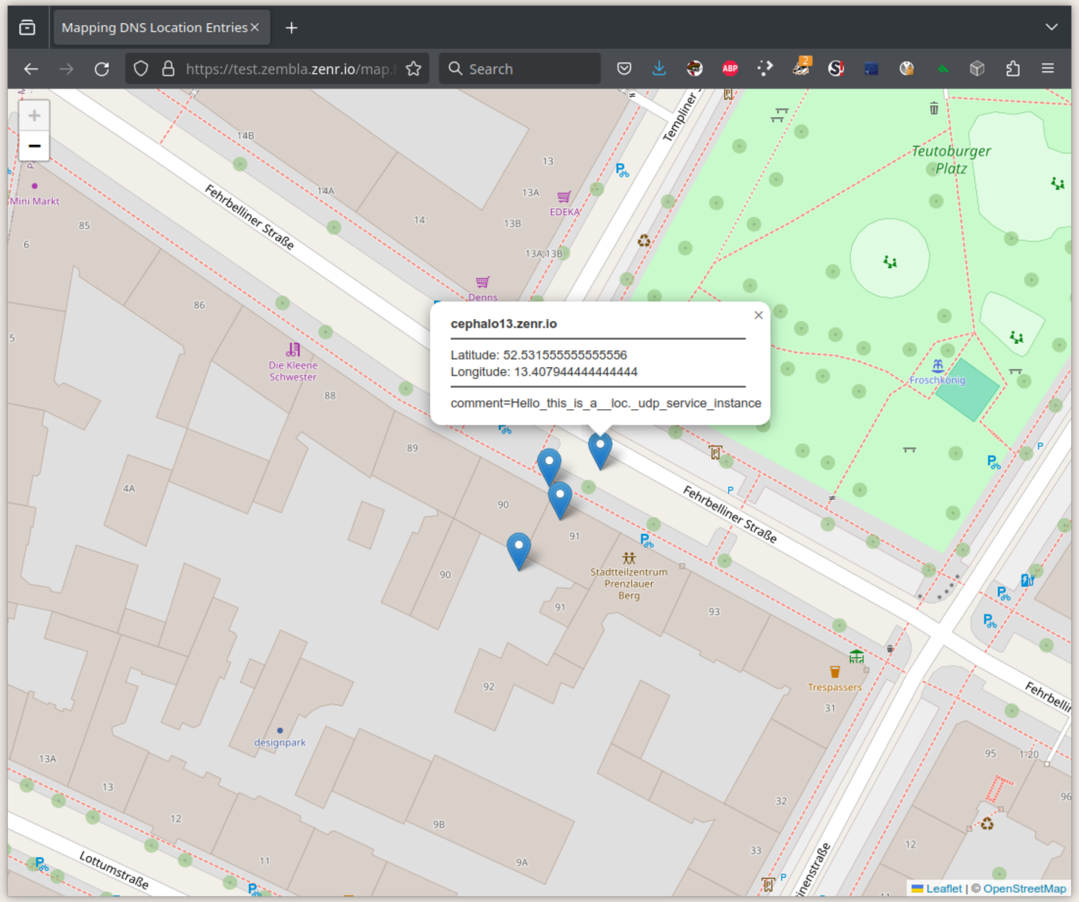
The dig or avahi-browse tools can be used to enumerate details of services of the service type _http._tcp – which act like shared bookmarks

dnssd-service offers configuration of sets of service types (PTR records) and service instance sets (SRV & TXT records).

It works with Linux, macOS & Win DNS-SD tools.

The service type PTR records allows each service type to be listed & used on demand. **dnssd-service** allows further DNS resource record types such as **PTR**, **SRV** and **TXT** resource records underneath the service type subdomain (eg `_http._tcp`) to determine sufficient information to access the resource, including IP address & port, together with any service specific details. See dns-sd.org for more examples.

Demo 1: group map update



cephalo13.zenr.io

Latitude: 52.531555555555556
Longitude: 13.407944444444444

comment=Hello_this_is_a__loc._udp_service_instance

dyn_loc is a tool allows the management of DNS LOC records under a KEY's FQDN. LOC records specify GPS locations. It currently supports real time position updates with supported platform tools: **termux** under Android & **gpsd** under Linux.

demo/map.html is a demo browser web app showing dynamic map of DNS-SD service type of location markers (**_loc._udp**).

This example shows a curated list created under zembla.zenr.io where each position marker is under control of their own LOC record under their own managed FQDN.

Further Work ...

Further work in progress:

- DNSSD web browser & management tool
- DNSSD wireguard service request and configuration provisioning workflow with tools allowing automated public IPv4/IPv6 allocation for freifunk hosted services – Freifunk hosted services made available the entire Internet in a decentralised method! :-)
- Consider further workflows with:
  - DANE, OPENPGP, SSHFP, TLDA
- Others?
  - Get Involved: be a pilot developer or user
  - Your feedback & ideas here .....

Prototype Fund

SPONSORED BY THE
Federal Ministry of Education and Research