To Name is to Own - Autonomous Name Spaces, Verifiable Identities, Assets and Services at the Edge

.ical Edit

08-15, 17:00–18:00 (Europe/Berlin), Bits&Bäume Workshops
Language: English

With a lightweight method of using the DNS trust root, decentralised communities can establish and self-manage secure hierarchies of trust for identities, assets and distributed services. This can be deployed outside the data centre monopoly and into the decentralised and local infrastructures at the network edge for complete autonomy with minimal resources and management. Why use blockchains with eternal history and ever increasing storage and resource requirements, when you can leverage and liberate the most highly distributed network service in existence?

This workshop offers participants the hands on opportunity to discover potential uses of dynamic DNS updates, authorised and signed utilising DNSSEC as a trust root & SIG(0) keypairs for update authentication.

- Learn how to configure and deploy an authoritative dynamic DNS service with DNSSEC enabled.
- Discover how to publish cryptographic SIG(0) public keys to delegate and update resource records.
- Configure secure wide area DNS service discovery to allow any device to securely publish services browsable over the global internet though avahi or dns-sd (DNS Wide Area Service Discovery).
- Securely delegate DNS subdomains to servers, even on a battery powered Raspberry Pi.

Workshop summary

Gentle Introduction

- DNS and how it works (usually does)
- DNS updates (DNS read/write capabilities)
- Wide Area unicast DNS-SD (global scope)
- DNSSEC process and details

Project sig0zonectl

- Stitching all the above together
- Github: **https://github.com/adam-burns/sig0zonectl**
- Please clone and explore during workshop

Further ideas and hacking

Workshop motivations

Autonomy

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

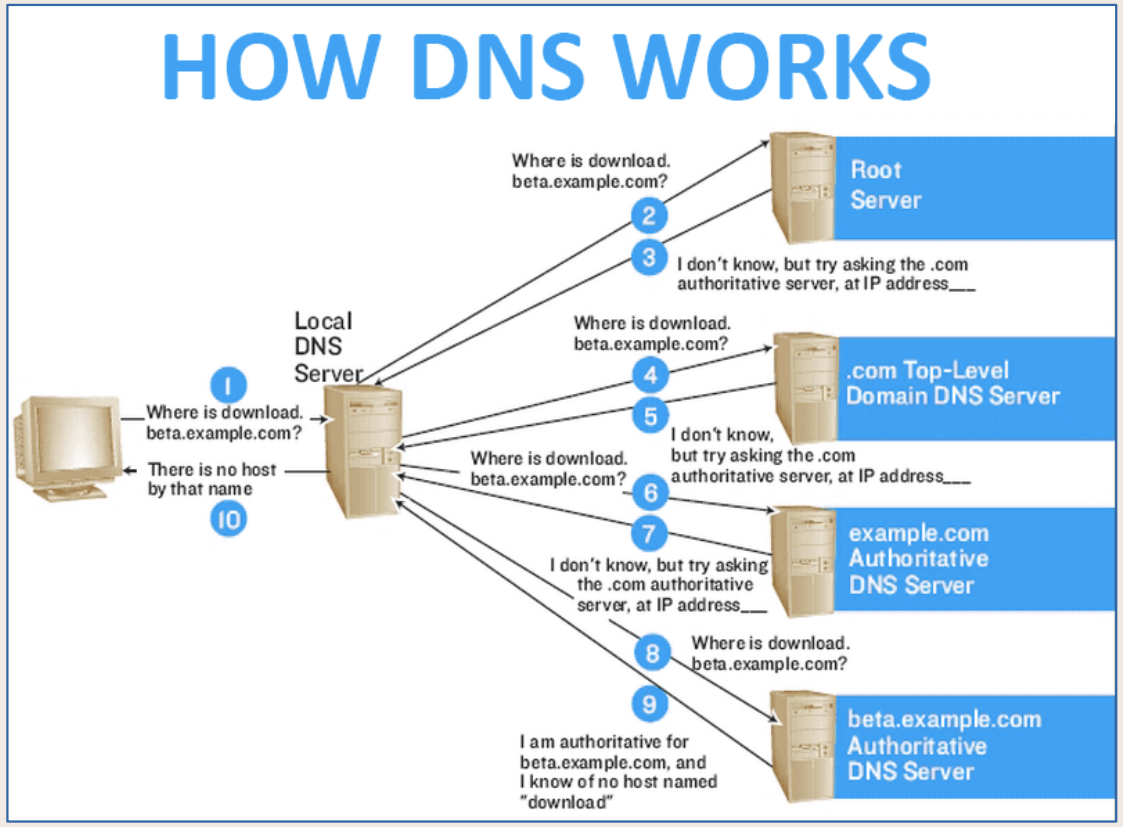
Sustainability

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

Politics

- growth of opportunity for local independent and community network service offerings
- remove unnecessary extractive 3rd party services
- remove unnecessary use of centralised data center infrastructures

Basic DNS: talkin’ bout a resolution …



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)

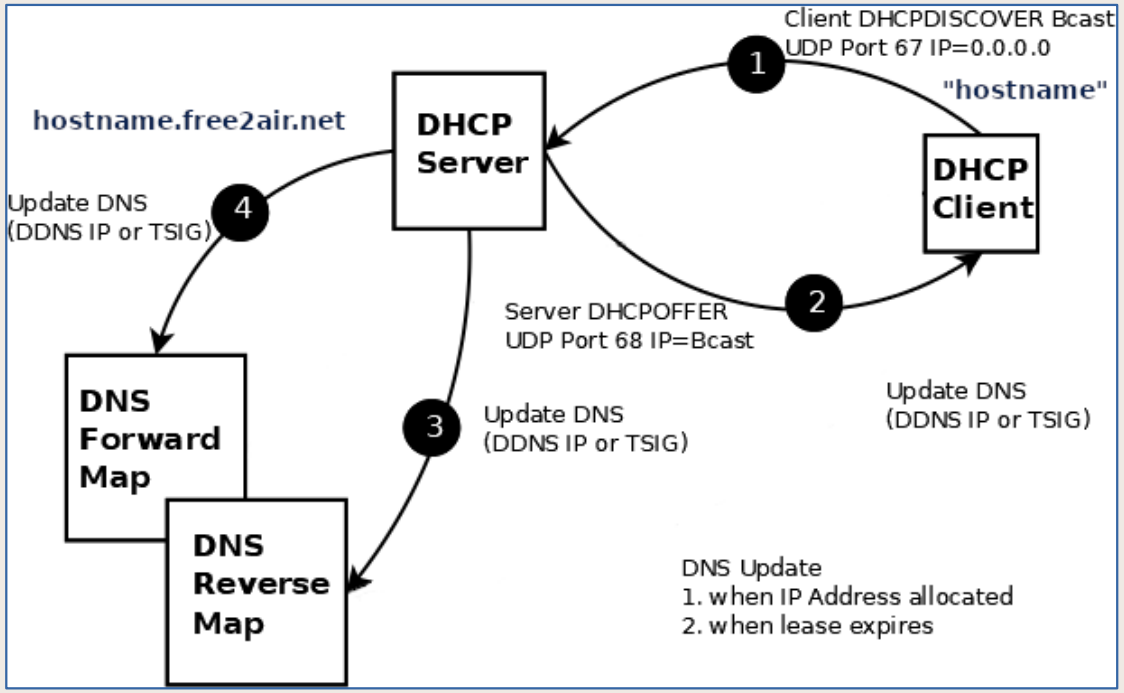
DNS transaction type usually a read-only ‘query’

- ask question (query)
- get answer (resolution)

DNS serves answers to queries to resource record types, e.g. **A records** for IPv4, **AAAA records** for IPv6 & **MX records** for mail server address resolution.

`nsupdate -T` gives full list of DNS resource record types.

Basic DNS: dynamic DNS update



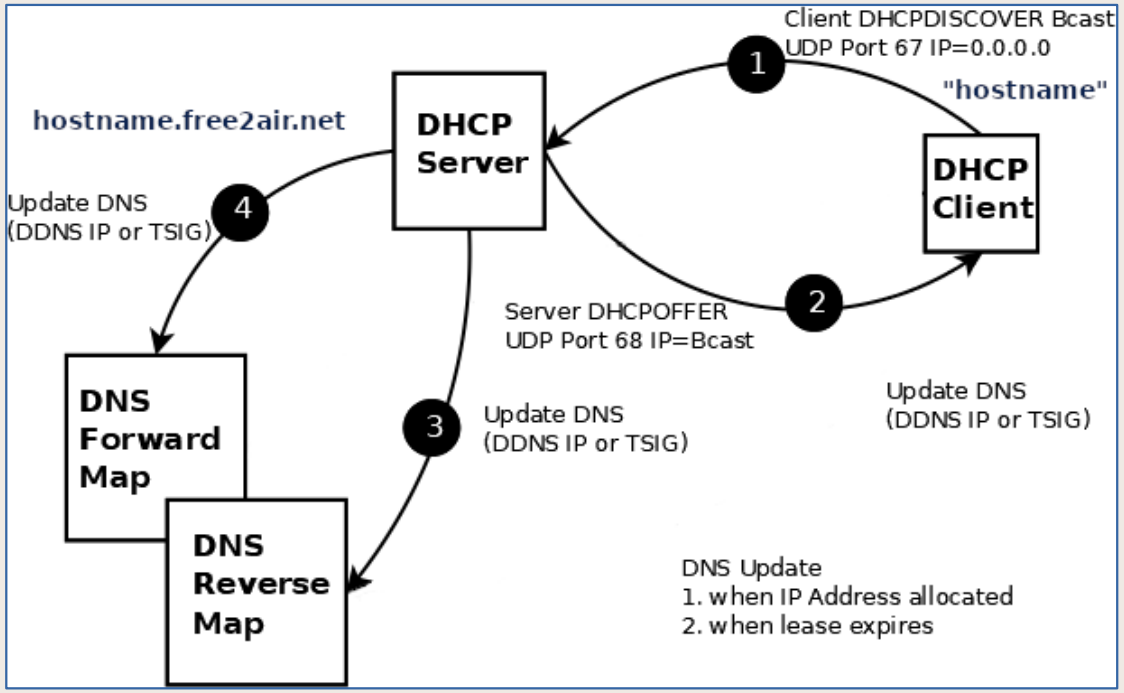
butchered from https://www.zytrax.com/books/dns/ch9/dhcp-1.png

Dynamic DNS:

- allows DNS records to be added, changed or deleted.
- uses DNS transaction type ‘update’.
- often deployed with DHCP servers for quick server name deployment for internal ‘trusted’ infrastructure.
- uses a shared secret authentication (TSIG), which
  - requires manual reconfiguration for new secrets.
  - is not very secure for 3rd party authentication at scale.

Rant: The shortcomings of TSIG encouraged some public DNS service providers to implement non-interoperable REST APIs over HTTPS to provide non-standard “dynamic DNS-like” services to their customers.

Example: Community Wireless Network

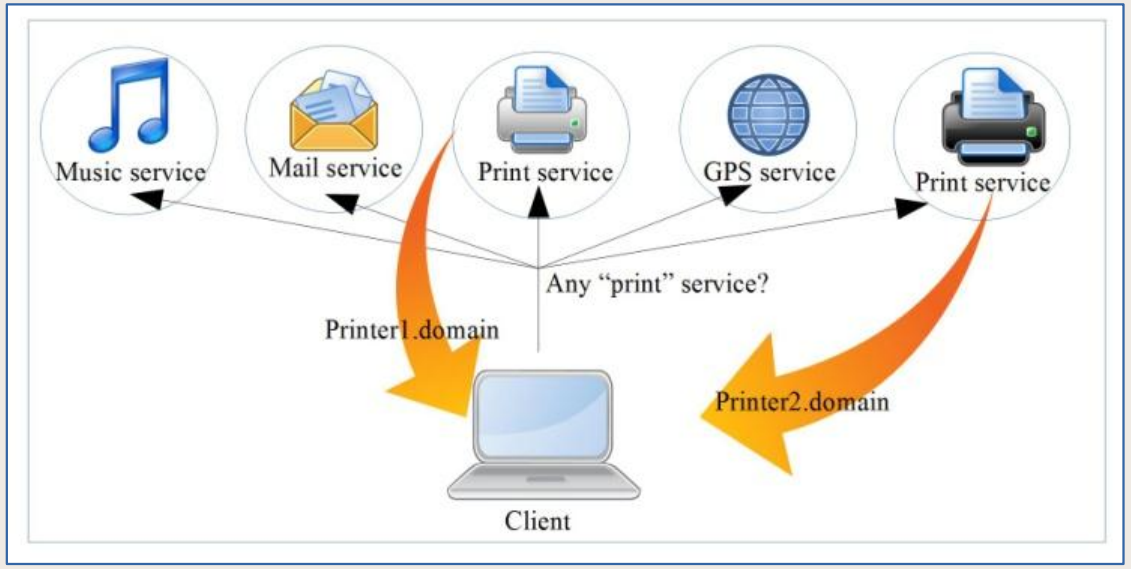


butchered from https://www.zytrax.com/books/dns/ch9/dhcp-1.png

```
# selected example of DHCP TSIG auto-deployed resource records
# DNS A records under free2air.net domain

Arifs-iPhone            A           93.97.42.65
Aude5-PC                A           81.187.145.171
BLACKBERRY-3199         A           93.97.42.66
CMAC                    A           81.187.145.166
FRANCESCA               A           93.97.42.65
Giless-iPhone           A           93.97.42.66
grrs-iPhone-7           A           93.97.42.71
iDizz                   A           93.97.42.69
iPod                    A           81.187.145.163
iPod-4                  A           81.187.145.166
james-iphone-2          A           93.97.42.66
JNR-iPhone              A           93.97.42.71
Lejito-s-iPhone         A           93.97.42.68
Livvys-iPhone           A           93.97.42.65
minint-pv12anv          A           93.97.42.67
Phone                   A           93.97.42.68
Riz                     A           93.97.42.66
robs-iPhone             A           93.97.42.67
#
Hards-iPhone            A           10.1.0.197
iPhone-2011-emb         A           10.1.0.247
```

DNS Service Discovery (DNS-SD)



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.domain.
b                       PTR         domain.
db                      PTR         domain.
dr                      PTR         domain.
lb                      PTR         domain.
r                       PTR         domain.
_services               PTR         _printer._tcp.domain.
$ORIGIN _printer._tcp.domain.
                        PTR         Printer1.domain
                        PTR         Printer2.domain
```

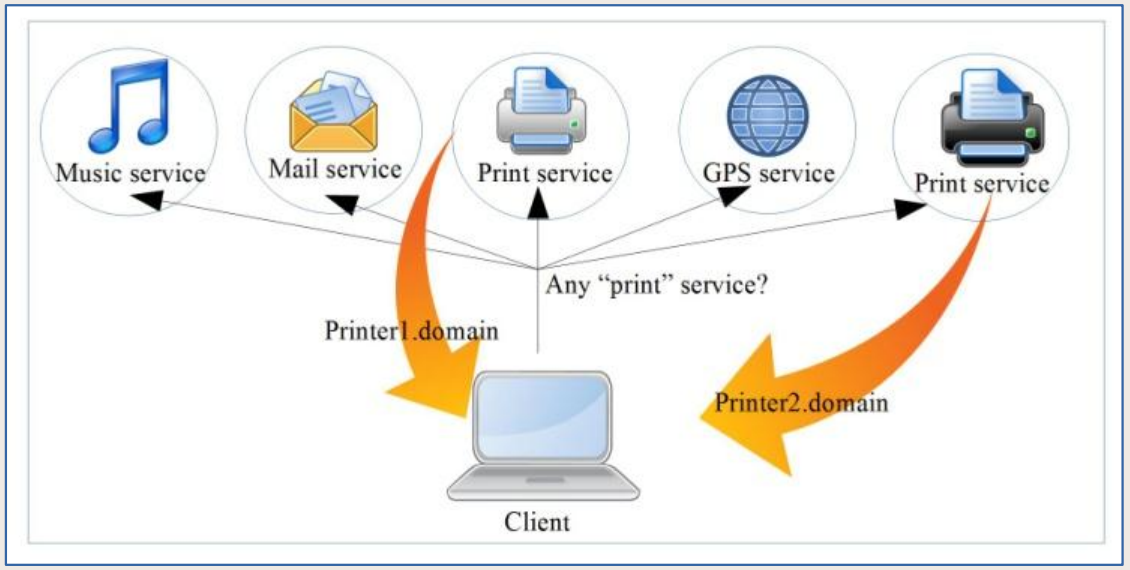
DNS-SD provides for 3 core functions within a DNS domain the use of DNS **PTR**, **TXT** & **SRV** resource records for:

- service registration (service offering)
- service enumeration (service 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)
- unicast DNS for global domains (for all other TLD addresses)

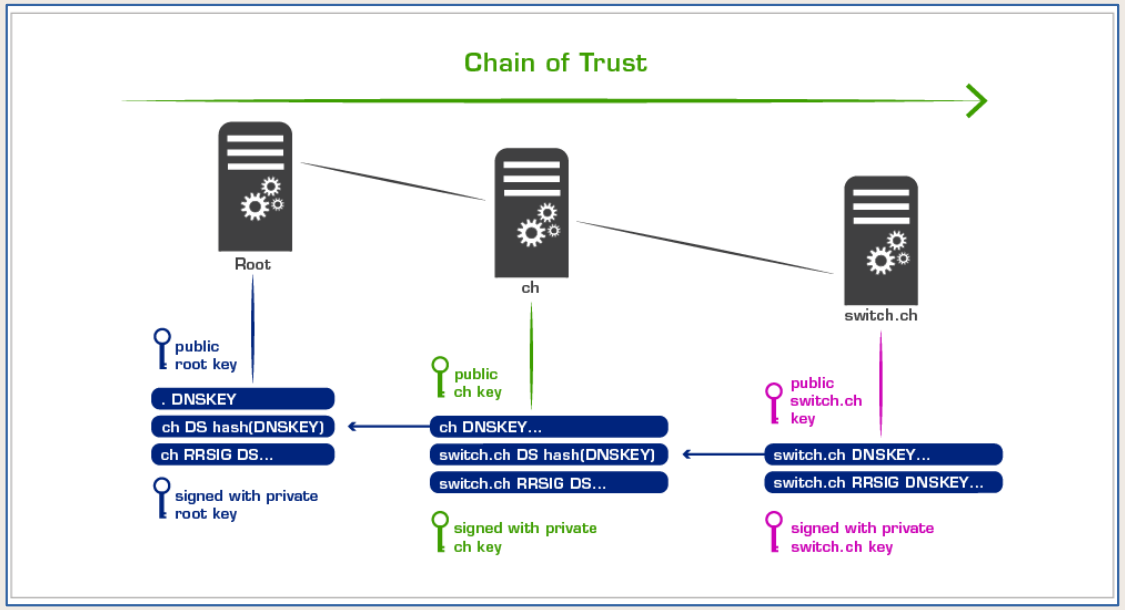
Example: Community Wireless DNS-SD



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

```
# small selection of DNS-SD browsable services
# and service instance examples
# under free2air.net domain
#
$ORIGIN _dns-sd._udp.free2air.net.
_services   PTR         _ftp._tcp.free2air.net.
            PTR         _ipp._tcp.free2air.net.
            PTR         _rfb._tcp.free2air.net.
            PTR         _ssh._tcp.free2air.net.
            PTR         _daap._tcp.free2air.net.
            PTR         _http._tcp.free2air.net.
            PTR         _printer._tcp.free2air.net.
            PTR         _sftp-ssh._tcp.free2air.net.
            PTR         _afpovertcp._tcp.free2air.net.
            PTR         _bittorrent._tcp.free2air.net.
            PTR         _pulse-sink._tcp.free2air.net.
            PTR         _workstation._tcp.free2air.net.
            PTR         _pulse-server._tcp.free2air.net.
            PTR         _pulse-source._tcp.free2air.net.
$ORIGIN _tcp.free2air.net.
_workstation PTR        media\032iii._workstation
_ssh         PTR        media\032iii._ssh
_rfb         PTR        RemoteGuest._rfb
$ORIGIN _rfb._tcp.free2air.net.
RemoteGuest  TXT        "fullname=" "type=shared" "u=guest"
```

Basic DNSSEC: DNS Chain of Trust

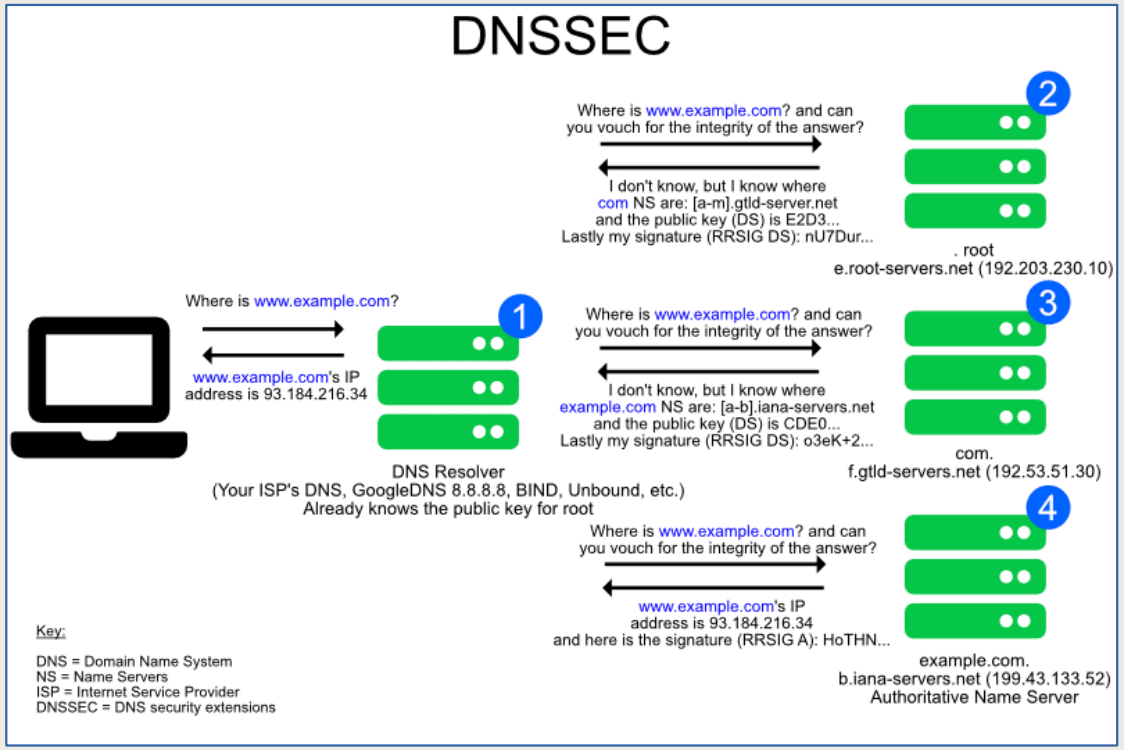


from https://www.nic.ch/export/shared/.content/images/dnssec_chain_en.png

DNS Security Extensions (DNSSEC) introduces a cryptographic chain of trust for integrity, with the **DNS root key** as the trust anchor.

The DNS root key signing ceremony is a regular event guided by an audited strong security process that brings together multiple independent stakeholders to update and sign the DNS root key.

Basic DNSSEC: in practice



from https://dnscrypt.info/_nuxt/img/DNS-DNSSEC.0de4990.png

DNSSEC adds new DNS resource record types:

- **DNSKEY records** for zone & key signing*.
- **DS records** for hashes of child zone DNSKEYs (linking chain of trust).
- **RRSIG records** for signatures of individual resource records.
- **NSEC records** for proof of non-existence of resource records within a signed zone.

Most major DNS server implementations (e.g. BIND9, Knot DNS) allow automated managed DNSSEC signing of even dynamic DNS zones(!)

* Early versions of DNSSEC used **KEY records** which is now only used for **SIG(0)** authentication.

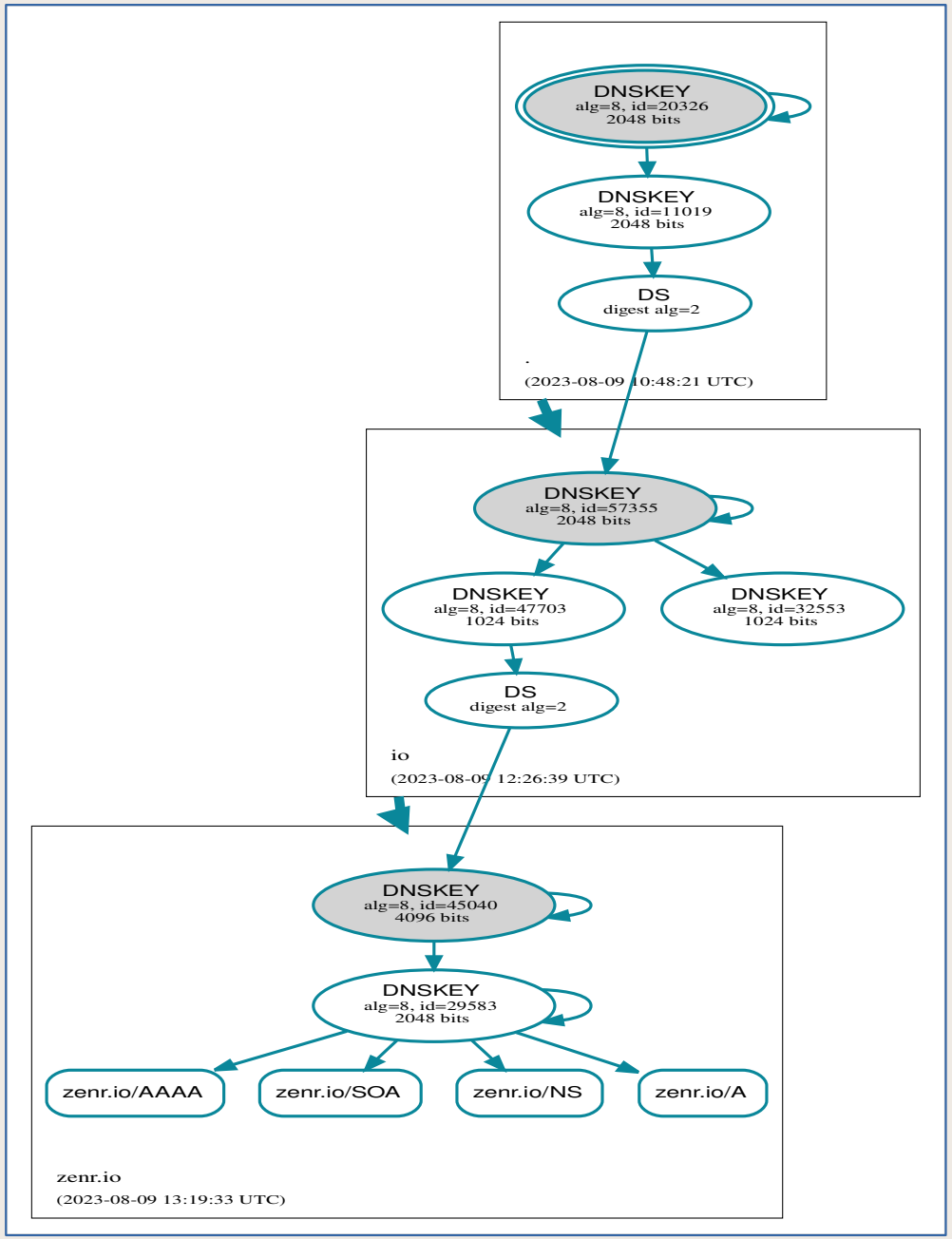
Example: Visualising DNSSEC chains of trust

The website **dnsviz.net** offers an on-line service of the open source dnsviz toolkit to visualise details of DNSSEC chains of trust.

For a selected domain, it uses:

- **DNSKEY records** to show zone key details.
- **DS records** to show link details in the chain of trust.
- **RRSIG records** & **NSEC records to** to show the set of signed resource record types within a signed zone.

Fun-with-DNS note: DNSSEC also supports IDN with punycode – for example, use dnsviz to visualise the trust chain of **xn--dr8h.zembla.zenr.io**.



Example visualisation of zenr.io trust chain from root

SIG(0) DNS Update 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 the KEY zebla.zenr.io

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

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

The private key should kept secure with the host that generates the key pair and submits the DNS update request.

SIG(0) Domain KEY Request

```
$ ZONE="zenr.io" NEW_SUBZONE="zembla" ./request_key
Generating key pair.
Kzembla.zenr.io.+015+55028
New SIG0 keypair for zembla.zenr.io generated in
./keystore
KEY request 'zembla._signal.zenr.io  IN KEY 512 3 15
duQIg/NgFjwsE8ZKUuXJUG2/NNFs4o4byuwnekT062U=' added
$
```

Using the sig0zonectl tool **request_key** to generate a domain KEY resource record

Using the sig0zonectl tool **request_key**, **zembla.zenr.io** is a KEY domain requested from the DNS server authoratitive for **zenr.io** zone.

```
$ dig +noall +answer +multiline zembla._signal.zenr.io KEY
zembla._signal.zenr.io.    60 IN KEY 512 3 15 (
         duQIg/NgFjwsE8ZKUuXJUG2/NNFs4o4byuwnekT062U=
                    ); alg = ED25519 ; key id = 23799
```

Using **dig** to get further information about the KEY zebla.zenr.io

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

The private key should kept secure with the host that generates the key pair and submits the DNS update request.

SIG(0) Domain KEY Acceptance

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

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

PROCESSING KEY REQUESTS

KEY 'zembla.zenr.io' submitted to add under zone 'zenr.io'
with '[key id = 23799]', IDN 'zembla.zenr.io'.
```

Using the sig0zonectl tool **process_requests** to publish new KEY zembla.zenr.io

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

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

PROCESSING KEY REQUESTS

KEY 'vortex.zenr.io' IS NOT submitted to add under zone
'zenr.io', as DNS resource records already exist.
```

Using The sig0zonectl tool **process_requests** denying KEY domain request zembla.zenr.io

With access to the **zenr.io** private KEY, the sig0zonectl tool **process_requests** grants subdomain KEY additions on a “first come, first served” (FCFS) policy basis.

If the subzone has no existing records at all, the KEY record is added and SIG(0) update access is granted for the KEY domain and all subdomains.

If any DNS resource record already exists, then the addition request made for publishing the KEY is denied.

DNS-SD domain configuration

```
$ ZONE="zenr.io" NEW_SUBZONE="zembla" ./dnssd-domain
$
```

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

DNS-SD offers standard DNS PTR resources to ease browsing & registering services within a domain.

```
$ORIGIN _dns-sd._udp.zembla.zenr.io.
b                       PTR     zembla.zenr.io.
lb                      PTR     zembla.zenr.io.
db                      PTR     zembla.zenr.io.
r                       PTR     zembla.zenr.io.
dr                      PTR     zembla.zenr.io.
```

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

For any particular domain, DNS_SD aware software query these PTR resource records underneath **`_dns-sd._udp`** to determine default domain hierarchies for service browsing and registration.

DNS-SD service instance registration

```
$ DNSSD_SERVICES="_ssh._tcp _http._tcp _ftp._tcp” \
ZONE="zenr.io" NEW_SUBZONE="zembla" ./dnssd-domain
$
```

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.zenr.io
...
```

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

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

The dig or avahi-browse tool can be used to enumerate details of one of the example _http._tcp service instance registered above

The DNS-SD standard offers a service instance set of DNS resource records with enough information to browse service types and enumerate service instance access details.

Each service type can be browsed & enumerated, DNS_SD 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.

DNS-SD service registration

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

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

PROCESSING KEY REQUESTS

KEY 'zembla.zenr.io' submitted to add under zone 'zenr.io'
with '[key id = 23799]', IDN 'zembla.zenr.io'.
```

Using the sig0zonectl tool **process_requests** to publish new KEY zembla.zenr.io

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

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

PROCESSING KEY REQUESTS

KEY 'vortex.zenr.io' IS NOT submitted to add under zone
'zenr.io', as DNS resource records already exist.
```

Using The sig0zonectl tool **process_requests** denying KEY domain request zembla.zenr.io

With access to the **zenr.io** private KEY, the sig0zonectl tool **process_requests** grants subdomain KEY additions on a “first come, first served” (FCFS) policy basis.

If the subzone has no existing records at all, the KEY record is added and SIG(0) update access is granted for the KEY domain and all subdomains.

If any DNS resource record already exists, then the addition request made for publishing the KEY is denied.

Future Work @ Workshop, Camp & beyond

Further hacks in order of difficulty:

- Simple DynIP style tool for IPv4 & IPv6 address updates (at camp?)
- Find my Phone style tool updating DNS **LOC** resource records for any named resource – not everyone's gear lives in a data centre (at camp)
- ansible script and/or docker-compose config to assist with zone on-device DNS zone delegation (WIP – still a manual configuration process).
- DANE browser implementation*
- Federated W3C DID identitity & verifiable credentials name spaces
- YOUR IDEAS HERE

* Ask me how angry I am about Mozilla and Google's 15 year refusal to implement DANE because it benefits them not to.