



RNDC and TSlG

Edward Lewis

lewis@tislabs.com





What is RNDCC?

- Remote Name Daemon Controller
- Command-line control of named daemon
- Usually on same host, can be across hosts





Configuring RNDP

- "rndc-conf" generates lines to be added to two files
 - ◆ named.conf
 - ◆ rndc.conf





Enabling RNDP in the server

- key definition

```
key rndc_key {  
    secret "dY7/uliR0fKGvi5z50+Q=="; algorithm  
    hmac-md5;  
};
```

- ◆ Warning: example secret looks good but is invalid (don't copy it!)

- controls statement

```
controls {  
    inet 127.0.0.1 port 953  
        allow { 127.0.0.1; }  
        keys { "rndc-key"; };  
};
```





Using an rndc.conf file

- /etc/rndc.conf specifies defaults for rndc
- E.g.,

```
key "rndc-key" {  
    algorithm hmac-md5;  
    secret "dY7/uIiR0fKGvi5z50+Q==";  
};  
  
options {  
    default-key "rndc-key";  
    default-server 127.0.0.1;  
    default-port 953;  
};
```





What can be done with RND

-
- `rndc stop` - kills server
 - `rndc status` - prints some information
 - `rndc stats` - generates stat file (named.stats)
 - `rndc reload` - refresh zone(s), w/variations
 - `rndc trace` - increases debug level
 - `rndc flush` - removes cached data
 - other commands in the ARM



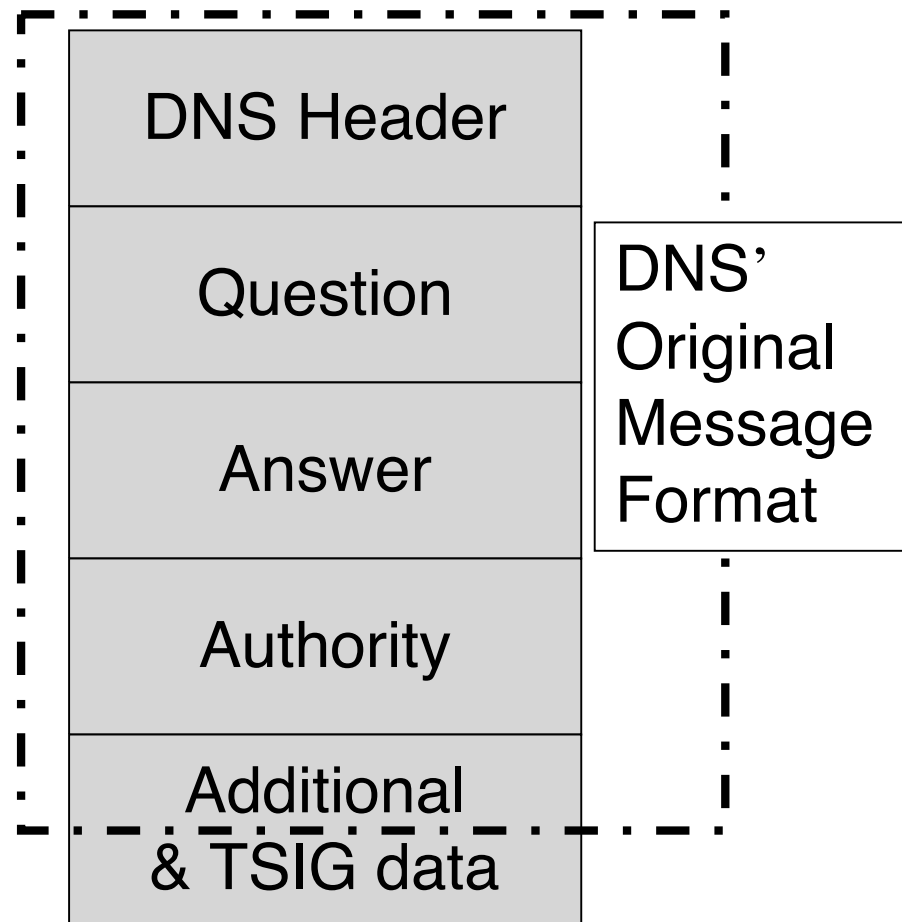


What is TSIG?

- A mechanism for protecting a message from a resolver to server and vice versa
- A keyed-hash is applied (like a digital signature) so recipient can verify message
- Based on a shared secret - both sender and receiver are configured with it



TSIG and Message Format





Names and Secrets

- TSIG name
 - ◆ A name is given to the key, the name is what is transmitted in the message (so receiver knows what key the sender used)
- TSIG secret value
 - ◆ A value determined during key generation
 - ◆ Usually seen in Base64 encoding
- 'Looks' like the rndc key
 - ◆ BIND uses same interface for TSIG and RNDC keys





Using TSIG to protect AXFR

- Deriving a secret
 - ◆ `dnssec-keygen -a ... -b ... -n... name`
- Configuring the key
 - ◆ in `named.conf` file, same syntax as for `rndc`
 - ◆ `key { algorithm ...; secret ...; }`
- Making use of the key
 - ◆ in `named.conf` file
 - ◆ `server x { key ...; }`
 - ◆ where 'x' is an IP number of the other server



Configuration Example

Primary server

10.33.40.46

```
key ns1-ns2.zone. {
    algorithm hmac-md5;
    secret "APlaceToBe";
};
server 10.33.40.35 {
    keys {ns1-ns2.zone.};
};
zone "my.zone.test." {
    type master;
    file...;
    allow-transfer {
        key ns1-ns2.zone.;
        key ns1-ns3.zone.;;
    };
};
```

Secondary server

10.33.40.35

```
key ns1-ns2.zone. {
    algorithm hmac-md5;
    secret "APlaceToBe";
};
server 10.33.40.46 {
    keys {ns1-ns2.zone.};
};
zone "my.zone.test." {
    type slave;
    file...;
    masters {10.33.40.46;};
    allow-transfer {
        key ns1-ns2.zone.;;
    };
};
```

Again, the secret looks okay, but is purposely invalid

TIME!!!

- TSIG is time sensitive - to stop replays
 - ◆ Message protection expires in 5 minutes
 - ◆ Make sure time is synchronized
 - ◆ For testing, set the time
 - ◆ In operations, (secure) NTP is needed



Other uses of TSIG

- TSIG was designed for other purposes
 - ◆ Protecting sensitive stub resolvers
 - ✦ This has proven hard to accomplish
 - ◆ Dynamic Update
 - ✦ Discussed later, securing this relies on TSIG



Alternatives to TSIG

- SIG (0)
 - ◆ Public key approach to same services
 - ◆ Has potential, but not much experience yet
- TKEY
 - ◆ Means to start with SIG(0) and wind up with TSIG
 - ◆ Also, Microsoft uses this with Kerberos via GSSAPI