Guidelines for Deploying DNSSEC

Best Practice Document

Produced by the CSC/Funet-led working group on AccessFunet

Authors: Antti Ristimäki (CSC/Funet), Tuukka Vainio (University of Turku), Kaisa Haapala (CSC/Funet)

August 2014

Document No:	GN3plus-NA3-T2-FN3.1
Version / date:	Version 1.0, 21 August 2014
Original language :	Finnish
Original title:	“Ohjeita DNSSEC-käyttöönottoon”
Original version / date:	Version 1.7, 12 February 2014
Contact:	antti.ristimaki@csc.fi

CSC/Funet bears responsibility for the content of this document. The work has been carried out by a CSC/Funet-led working group AccessFunet.

The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 605243, relating to the project 'Multi-Gigabit European Research and Education Network and Associated Services (GN3plus)'.

Table of Contents

Executive Summary 1

1 What is DNSSEC? 2
- 1.1 Using DNSSEC Validation 2
  - 1.1.1 Name server software 2
  - 1.1.2 Time synchronisation 3
  - 1.1.3 Trust anchor 3
  - 1.1.4 Firewalls 4
  - 1.1.5 Monitoring 4
- 1.2 DNSSEC signatures for your own zones 5
  - 1.2.1 Signing system architecture 5
  - 1.2.2 Signing key management 6
  - 1.2.3 Choosing parameters 7
  - 1.2.4 Updating DS records 8
  - 1.2.5 Monitoring 9

References 10

Glossary 11

Executive Summary

This document contains guidelines and best practices for deployment, administration and monitoring of DNSSEC. These guidelines and recommendations are based on publicly available documentation (such as RFC documents) and CSC/Funet's own experiences and observations. However, it should be noted that not all of these recommendations are necessarily suitable for every environment. Each environment's particular technical limitations must always be taken into consideration, as should practices relating to individual operating models.

This document does not aim to give a detailed description of domain name service or DNSSEC operation principles. Instead it gives a very brief introduction to DNSSEC. More detailed information can be found elsewhere, mainly in related RFC documents.

1 What is DNSSEC?

Domain Name System Security Extensions (DNSSEC) can be used to ensure that a DNS response originates from the correct authoritative name server, and that the response has not changed during transmission. Unlike original DNS protocol, it provides highly effective protection against DNS spoofing attacks, in particular Man-in-the-Middle attacks. Response validation is based on public-key cryptography. DNS records are signed with the zone owner's private key and verified using the corresponding public key. The private part of the signing key is usually known only to the zone's owner/administrator/maintainer, while the public part is published in DNS as a DNSKEY record.

The validation of DNSSEC signatures (RRSIG records) is based on chains of trust that follow the DNS hierarchy. In practice, what happens is that each zone's public key (or more specifically, a cryptographic digest of it, known as the DS record) is signed by the zone key at the higher level in the DNS hierarchy. The Internet root zone's public key is typically found at the top of a trust chain. Trust chains consist of a series of DS records that contain hash digests generated from public keys. These DS records are published one level up in the DNS hierarchy and are signed with the parent zone's key.

DNS has proved to be an extremely functional and, above all, scalable database, and DNSSEC signatures enable the publication of not only traditional name server information but also more sensitive data. For example, it is a good idea to publish SSH key fingerprints in DNS, if the fingerprint records (SSHFP records) can be signed and validated using DNSSEC.

1.1 Using DNSSEC Validation

Using DNSSEC validation, a DNS resolver can use signatures to authenticate the origin of the DNS records that it has received and verify their integrity. This section explains how to start using DNSSEC validation on a resolving name server and which issues require particular attention.

1.1.1 Name server software

In order for a resolver to be able to use DNSSEC validation, you must install name server software that supports DNSSEC validation. Bind and Unbound are known to support DNSSEC validation, if a sufficiently up-to-date version is used. (Version 9.7.5 or later is recommended for BIND, and 1.4.16 or later for Unbound.) It is generally better to use a popular name server software, as support will usually be available from user communities. As DNSSEC is still a relatively new technology in production, you can expect to encounter

DNSSEC-specific issues with name server software, as well as vulnerabilities. It is therefore vital to follow bug and vulnerability reports for the name server software in use.

1.1.2 Time synchronisation

The validity of DNSSEC signatures always has an explicit start and end time, so it is vital that the validating resolver's internal clock is set to the correct time. If the resolver's time is incorrect, it can interpret valid signatures as expired and vice versa. It is therefore recommended that you synchronise your validating resolver to a reliable time source. For example, you can use NTP and/or PTP protocols to synchronise your resolver's time. It is a good idea to configure NTP servers with an IP address rather than the NTP servers' DNS names, as this will avoid circular dependencies. NTP servers' DNS names will not be resolved if they are signed and the resolver's clock is set to the incorrect time.

As synchronising the time on virtual machines can be problematic, it is recommended that dedicated physical server equipment is used for your resolver server, if possible. As the domain name service is a critical part of network infrastructure, we do not recommend implementing it using a virtual server dependent on a virtualisation platform.

The validating resolver informs the authoritative name server of its ability to receive DNSSEC records using a separate EDNS0 record and, specifically, with the DNSSEC OK (DO) bit it contains. Using this EDNS0 record, the resolver can also inform the authoritative name server that it is able to receive larger UDP responses than defined in the original DNS standard (DNSSEC increases the size of responses so that they generally exceed 512 bytes). The maximum size of UDP responses is usually defined in the name server configuration (with Unbound "edns-buffer-size" and with Bind "edns-udp-size") and the recommended value is 4,096 bytes according to RFC 6891 [RFC 6891]. This also tends to be the default value in software. If the resolver's firewall blocks IP fragments, it is worth configuring the maximum size of UDP responses to a value smaller than the MTU generally used in Ethernet networks (1,500 bytes), to 1,480 bytes, for example, to reduce the likelihood of fragmentation You can read more about the effects of DNSSEC on the resolver's firewall later in this document.

1.1.3 Trust anchor

DNSSEC validation is based on chains of trust, which is why the validating name server must be configured with a 'trust anchor' to act as the first link in the trust chain. The trust anchor is either the public part of a signed zone's key (a DNSKEY record) or a cryptographic hash digest of it (a DS record). As the Internet's root zone is signed, we recommended that you use the root zone's public key or its digest as your trust anchor. The root's trust anchor can be retrieved from IANA's website and, before use, its authenticity should be verified using some other mechanism than DNS(SEC), for example, the PGP web of trust. It is also a good idea to ensure that the trust anchor taken from IANA's website corresponds to the root zone's public key found in the name service. The root's DNSKEY record can be transformed into a DS digest using the ldns-key2ds tool found in the LDNS software suite (for example,'dig . dnskey > root.dnskey && ldns-key2ds -2 -n root.dnskey').

At the time of writing, it was not known when the root zone's public key will be rolled over, but sooner or later this will happen. When this happens, the validating resolvers' trust anchors must also be updated. RFC 5011 describes a mechanism that a validating resolver can use to update its trust anchor automatically. As it is not

yet known when and, in particular, how the root zone's key will be renewed and how the aforementioned RFC 5011 mechanism will work in practice, the root zone's key change must also be monitored in other ways. We recommended that you set up a monitoring tool that will notify the validating resolver's administrator when the root key changes. The simplest way to implement this type of tool is as a shell script run under cron.

1.1.4 Firewalls

DNSSEC records significantly increase the size of DNS responses, which is why they may not fit in the 512-byte UDP packets defined in the original DNS standard. Name servers use a special EDNS0 extension to signal their ability to receive UDP response messages larger than 512 bytes. If the UDP response's maximum size (as signalled by the resolver using EDNS0) is larger than the MTU (Maximum Transmission Unit) of the link, the authoritative name server must send a fragmented response. This is why it is important for the resolver's firewall to be configured to accept UDP fragments.

If the response message exceeds the size of the maximum UDP response signalled using EDNS0, the authoritative name server will return a truncated response using UDP with the 'Truncated' flag set. The Truncated flag will cause the resolver to re-query using TCP. Situations like this are more common when using DNSSEC, so it is important that the resolver's firewall also allows DNS queries using TCP.

If the resolver server is behind a firewall, it is also important to ensure that it doesn't discard the ICMP error messages required by Path MTU Discovery (PMTUD). This is especially important for IPv6, as IPv6 routers do not perform packet fragmentation. Instead, they drop oversized packets and send the originating host an ICMP error message.

1.1.5 Monitoring

DNSSEC introduces considerable changes to the operation of the domain name service, and can also generate new kinds of problems. It is therefore vital to monitor the operation of DNSSEC and its core subsystems. With regard to DNSSEC validation, at least the following things should be monitored:

- The number of validation errors and names that fail validation should be analysed when necessary, for example, by increasing the logging verbosity of the name server software.
  - The availability of resolver statistics (including validation errors) depends on the name server software used. WithUnbound, for example, you can dump statistics using the 'unbound-control-stats' command.
- It requires careful consideration whether or not you want validation errors to be logged continuously. **Note that continual comprehensive logging can, at worst, load the resolver heavily and thereby open up a new attack vector**. In Funet's validating resolvers continuous logging is usually switched off and only turned on temporarily as required (if there is a need to analyse the reason for validation errors, for example).
- The number of SERVFAIL responses.
- The resolvers' time synchronisation status (for example, NTP synchronisation).

It is vital for you to be able to detect any sudden and significant rise in the number of validation errors as quickly as possible. When setting up a monitoring system for validation errors, the alert thresholds should be defined after a careful analysis, so that a continual stream of validation errors does not cause false positives and, in the worst-case scenario, hide serious problems. In the case of Funet, the threshold values set for the monitoring of Funet's resolver service will trigger an alarm when validation errors exceed a rate of one per second (under normal operation, the number of validation errors is less than one per minute).

1.2 DNSSEC signatures for your own zones

1.2.1 Signing system architecture

In most signing systems, the signing server is logically located between the administration system of DNS data and the public name server. The signing system is then a sort of 'bump in the wire' component that has no external public interfaces visible to the Internet. In practice, the signing system receives unsigned zone files from the administration system using, for example, an AXFR/IXRF ('zone transfer') or some other interface (such as SSH/SCP). The signing system then signs the zone files and transfers the signed versions of the zone files to a public name server – using one of the previously mentioned interfaces, for example. If the administration system of DNS data supports DNSSEC signing, it can be used for signing, which will significantly simplify the system architecture. It is not, however, wise to implement the signing function using a public name server, as private signing keys are more challenging to protect than if key storage and signing occurs within a system that has no external interfaces.

Your choice of software will be one of the key factors that affects the architecture of your signing system. OpenDNSSEC, which is based on open source code, is in widespread use and is even used to sign many Top Level Domain (TLD) zones – it can therefore be recommended. OpenDNSSEC also has an extremely active user and developer community, which generates considerable added value compared to closed commercial solutions. The latest version of Bind name server software also has support for so called inline signing function [Inline-signing], but there was no information on its practical functionality available at the time of writing. However, this would make it possible to use a BIND name server as a hidden master server through which unsigned zone files are transferred as signed files to a public name server.

It is recommended to implement redundancy to the signing system at least to the extent that under a failure, the signing functionality and the private signing keys can be manually migrated to a backup system within a reasonable time. The authors do not recommend using an automatic high availability feature, as DNSSEC involves many aspects that should be checked before a backup system is activated. Before activating a backup system, it is important to ensure that the backup signing server uses the same keys as the primary system, and that it has the same state information with regard to key rollover timing and other signing parameters. Regarding the importance of the signing system redundancy, it is worth remembering that a fault in the signing system will not in itself necessarily make the affected DNS domains unreachable, but will prevent zone updates. As a result, DNSSEC signatures will not update either and can expire if the signing system is not fixed quickly enough. By choosing the signing parameters wisely, one can influence how critical the availability of the signing system is – there is more on this subject later in this document.

When building redundancy to the signing system, one should remember to ensure that the primary and the backup systems always use the same keys. If the keys are pre-generated, they can be copied from the primary

server to the backup server during the signing system initialization. However, if keys are generated dynamically, one must make sure to synchronise the primary and backup systems. The metadata used to sign zones must also be synchronised between the servers. This metadata includes information on the timing of key rollovers. In OpenDNSSEC, for example, this data is located in a Key and Signing Policy (KASP) database, whose content must also be synchronized between the servers.

1.2.2 Signing key management

As DNSSEC is based on public-key cryptography, the reliability of the private key used to sign DNS records forms the foundation of the entire system. When designing a signing system, one should pay particular attention to the generation and secure storage of the private keys. When generating keys, particular attention should be paid to ensuring that keys are as random as possible, making them hard to crack with brute force. You can increase the randomness of your keys by, for example, using an external random number generator. These are available as, for example, USB memory sticks. It is also possible to utilize the RdRand random number generator included in the latest Intel processors – it essentially feeds entropy into the operating system's entropy pool. When generating signing keys, you shouldn't use Linux servers' /dev/urandom files as a source of entropy, as the random numbers they generate will not necessarily be random enough.

Signing keys – and their private parts in particular – should be stored as carefully and with as much protection as possible to ensure that no outsiders can gain access to them. If possible, keys should always be encrypted during storage in, for example, an encrypted file system. The downside of this is that, when booting the signing server, the administrator must enter the password used to protect the key storage file system. On the other hand, this can prevent the misuse of keys by, for example, stealing a hard disk. Keys can also be stored on a separate HSM (Hardware Security Module) device. However, these are usually quite expensive and require some degree of special expertise. Although there is good reason to use HSM devices to sign and store keys for zones in the highest levels, such as country-specific ccTLD zones, the use of HSM devices for signing lower-level zones is not necessary if the secure storage and randomness of keys can be guaranteed in other ways. A list of OpenDNSSEC-compatible HSM devices is available [HSM_devices], together with information on choosing an HSM device.

You must also remember to make backup copies of signing keys, preferably so that the backups are not stored in plain text format but are encrypted, for example using PGP. If your signing software allows, it is a good idea to completely prevent the use of keys that have not been backed up yet. For example, OpenDNSSEC can be configured so that it will not activate a key unless it has been backed up. If keys are dynamically generated when required, it is worth trying to automate the backup process as well, so that whenever the signing software generates new keys, that batch of keys is backed up. Another option would be to pre-generate – and back up – a large number of keys in advance when initialising the signing system.

When this document was written (in October 2013), the RSA algorithm appeared to be the surest bet for generating signing keys, as it can be considered both secure and well supported. 2,048 bits is the recommended length for RSA keys. There is no need for a stronger RSA key, as the Internet's root zone is signed using a 2,048-bit key. If zone signing is extremely slow using a 2,048-bit RSA key, one can also use a separate 1,024-bit RSA key as a Zone Signing Key given that it is rolled over more frequently than the zone Key Signing Key. When it comes to key algorithms, it is also recommended to ensure that the components of the signing system (software, auxiliary devices, etc.) also support keys based on elliptical curve algorithms (ECDSA keys), which will probably become more widespread in the future.

To be precise, DNS records are not signed as is – a cryptographic digest is first calculated and then signed. To some extent, the algorithm used to calculate the digest affects how computationally easy or difficult it is to crack the signature. We recommend that you use the strongest possible digest algorithm – at least SHA-256. We definitely do not recommend the use of the MD5 digest algorithm, due to its known vulnerabilities.

The rollover of keys at regular intervals reduces the likelihood of computational cracking. To make key rollover easier, we recommend the use of two different key pairs: A Key Signing Key (KSK) used to sign only zone keys (the DNSKEY RRset), and a Zone Signing Key (ZSK) used to sign the zone's actual DNS records. Rolling KSK keys requires interaction with the parent zone, whilst ZSK keys can be rolled without touching the parent zone. It is definitely a good idea to leave the rollover of ZSK keys to the signing software. Four times per year is a sufficient rollover interval for ZSK keys. There are two schools of thought when it comes to the rollover of KSK keys. Some think that it is worth rolling KSK keys only when the signing system/algorithm is renewed, while others think that renewing KSK keys is the only way to remember the rollover process, and should therefore be practised regularly. If you want to roll your KSK keys on a regular basis, an interval of 1–2 years is considered suitable. If you do not want to roll your KSK keys regularly, it would be sensible to use a stronger key, for example, 4,096 bits.

1.2.3 Choosing parameters

RFC 6781 [RFC 6781] and RFC draft draft-ietf-dnsop-dnssec-key-timing [Key-timing] give good recommendations for choosing DNSSEC signing and timing parameters. When setting up a DNSSEC signing system, it is a good idea to read the above-mentioned documents carefully. However, this section will highlight a few of the most important points.

The key thing to remember when choosing timing parameters is to ensure that, if an error occurs, the administrator has enough time to detect and fix the problem before the zone's signatures start to expire. The choice of parameters has a direct effect on how long you have time to detect and fix any errors that occur in the signing system or elsewhere in the signing process chain. If the default validity period for signatures is for example only two days, then the signing system can be down for only two days, until the signatures start to expire. The signature renewal interval also has an effect on signature expiration. If, for example, signatures are always renewed at least 10 days before they expire, then the freshly signed zone will never contain any signatures that will expire in less than ten days.

With regard to timing parameters, it is important to consider how quickly administrators can fix a problem, taking weekends and holidays into account. A reasonably safe setting is, for example, that signatures are always valid for 14 days and are renewed at the latest when they have 10 days of validity remaining. In practice, this gives 10 days to detect and fix a fault that prevents updates to a zone and, in particular, its signatures. Of course, you must also remember how the frequency of the signing cycle affects the system: if the signing process is launched only once a day and you are using 10-day limits for signatures, a zone can, just before signing, contain signatures that will expire in nine days. It is usually sensible to sign a zone much more frequently than this, for example once an hour, as it is then also much easier to monitor that the entire signing process chain works. In addition to signing at regular, defined intervals, the zone should naturally also be signed when information is changed. Regular signing is carried out to ensure that, even if the zone's content remains intact, the signing process will be launched at regular intervals (including going through the resource records that need to be signed, evaluating the need to roll keys, etc.), thus keeping the zone's signatures refreshed.

DNSSEC also has an effect on the values contained in a zone's SOA record, as the SOA record tells slave servers how often they need to check their master server, whether the zone has been updated and – if the master server cannot be reached – how long it is before they should interpret a zone as expired. DNSSEC affects the choice of these values, as it is preferred to let the zone expire from an authoritative name server than to return expired signatures. If a zone expires, the authoritative name server returns a SERVFAIL response to the resolver, and the resolver should query another authoritative server. However, if a zone does not expire before the signatures expire, the authoritative server returns an expired signature to the resolver. The validating resolver then naturally interprets this as incorrect and returns a SERVFAIL response to the original requester and, depending on the type of original requester (a forwarding name server versus the resolver library in a user's operating system), the latter may give up at this point. We therefore recommend that you set your zone's SOA expiry timer to a value that is an order of magnitude lower than the validity period of your signatures.

DNSSEC also enables authenticated denial of existence. In practice, this means information that a requested name or record does not exist. This is done with the aid of specific NSEC records. However, NSEC records have a negative side effect – iterating through the zone NSEC records it is trivial to enumerate the contents of the zone. Although all DNS data is by default public, the ability to list entire zone's data can sometimes be detrimental. It is therefore wise to use hashed NSEC3 records instead of NSEC records. One exception is for zones that contain only a few public records (such as only a www record or CNAME). In such cases, using NSEC3 does not generate any added value compared to NSEC. NSEC3 also has an OptOut feature, but it is only really relevant for large delegation-centric zones. There is no need to use OptOut when signing a typical second-level zone (such as funet.fi).

1.2.4 Updating DS records

In order to be able to validate a signed zone, the DS record corresponding to the zone's KSK key must be published in the parent zone and signed using the parent zone's (ZSK) key. For example, domain names ending in .fi, must publish their DS records in the 'fi' root, which is done via the Finnish Communications Regulatory Authority's domain name system [fi_domains]. DNS service providers can automate the submission of DS records using the Web Service interface, but end users must manually update their DS records through an online web interface. However, as this needs to be done only at the launch phase and when rolling KSK keys, it is not expected to cause unnecessary work load.

When it comes to reverse zones, the parent zone is typically a zone administered by the local internet registry. For Funet's customers' reverse zones, the parent zone is thus usually under Funet's administration. Exceptions include PA or legacy address spaces, where the corresponding reverse zone's parent zone is usually a reverse zone directly administered by the corresponding regional internet registry (RIR, for example, RIPE or Arin). At the time of writing (October 2013), Funet's reverse zones were not yet signed, which is why there is as yet no interface to update DS records. If the parent zone is directly administered by, for example, an RIR, its DS records are updated via the interface provided by the RIR. RIPE's DS records can be updated via the Webupdates interface [RIPE_updates] or via an automatic email interface.

1.2.5 Monitoring

If monitoring is an important aspect of DNSSEC validation, it is even more important for signed zones in order to ensure their accessibility. The following things should be monitored for a DNSSEC-signed zone:

- The zone can be DNSSEC validated. If the resolver used by the monitoring system supports DNSSEC validation, this check can easily be implemented by setting the DO bit in a DNS query and checking that the AD (Authenticated Data) bit is set in the response. The exact technical solution used to implement the check depends on the monitoring system used, but a broad range of DNSSEC extensions are available for Nagios, for example.
- Remaining validity of signatures. During normal operation, the remaining time should resemble a sawtooth diagram: it gets smaller as it approaches the signature refresh threshold (for example, 10 days) and jumps back to the defined validity period (for example, 14 days) when the signature is renewed.
- The zone's last update: an unsigned zone can go for years without updates, but a signed zone should be updated on a regular basis, to ensure that its signatures do not expire. The update interval is defined by the parameters set, but when they are known, you should monitor the zone to ensure that the last update was not longer ago than it should have been. Monitoring logic can be implemented using, for example, the following:
  - If a Unix timestamp is used as an SOA serial number, you can obtain the exact time of the last update directly from the SOA serial number.
  - If your SOA serial number is in the datecounter format YYYYMMDDXX, you will not be able to see the exact time of the last update directly. But since the SOA record and also its signature updates with every change in the zone, the inception timestamp of the SOA record's signature will show the exact time of the last update. However, you should note that the signing system may have a safety interval in timestamps for possible clock skew. For example, OpenDNSSEC by default defines the signature inception timestamp (the starting time of the signature's validity period) as [T_{now} - 3600s]
- Zone transfers between the signing system and the public name server (or between a public master server and its slave servers). It is essential to note if the zone doesn't update quickly enough to some authoritative name server. If the zone's last update is monitored as described in the previous point, all you have to do is check that all authoritative name servers have the same version of the zone.

It is also important to make a couple of checks after signing a zone before updating it to the public name servers. It is particularly important to check that a signed zone's records are validated against the zone's own keys (self-validation) and that the zone's keys are signed using a KSK key for which a corresponding DS record is found in the parent zone. For example, it is quite straightforward to implement these checks in OpenDNSSEC by defining the NotifyCommand configuration option, which will trigger the specified check script. In addition or in part to implement the above-mentioned checks, you can use readily available tools such as [validns].

References

Related RFC documents

[RFC 4033]	RFC 4033: DNS Security Introduction and Requirements http://tools.ietf.org/html/rfc4033
[RFC 4034]	RFC 4034: Resource Records for the DNS Security Extensions http://tools.ietf.org/html/rfc4034
[RFC 4035]	RFC 4035: Protocol Modifications for the DNS Security Extensions http://tools.ietf.org/html/rfc4035
[RFC 6871]	RFC 6781, DNSSEC Operational Practices, Version 2 http://tools.ietf.org/html/rfc6781
[RFC 6891]	RFC 6891, Extension Mechanisms for DNS (EDNS(0)) http://tools.ietf.org/html/rfc6891

Other references

[DNS-OARC]	OARC's DNS reply size test server https://www.dns-oarc.net/oarc/services/replysizetest
[DNStest_fi]	http://dnstest.ficora.fi/frontpage.php?lang=en
[fi_domains]	https://domain.fi/info/en/index.html
[HSM_Devices]	https://wiki.opendnssec.org/display/DOCREF/HSM
[Inline-signing]	https://kb.isc.org/article/AA-00626/
[Key-timing]	DNSSEC Key Timing Considerations draft-ietf-dnsop-dnssec-key-timing-03.txt http://tools.ietf.org/html/draft-ietf-dnsop-dnssec-key-timing-03
[RIPE_updates]	https://apps.db.ripe.net/webupdates
[validns]	http://www.validns.net/
[Verisign_debug]	http://dnssec-debugger.verisignlabs.com/

Glossary

DNS	Domain Name System
DNSSEC	Domain Name System Security Extensions
ECDSA	Elliptic Curve Digital Signature Algorithm. The first set of extensions was published in 1999 by the Internet Engineering Task Force as RFC 2671, also known as EDNS0
EDNS0	Extension mechanism for DNS. T
IANA	Internet Assigned Numbers Authority, a department of ICANN that manages DNS root zones
ICMP	Internet Control Message Protocol
NSEC	Next SECure record. A DNS record type used to prove a name does not exist in the DNS.
NTP	Network Time Protocol
PGP	Pretty Good Privacy. PGP software uses a “web of trust”, in contrast to the X.509 system which is based on a certificate authority.
PMTUD	Path MTU (Maximum Transmission Unit) Discovery
PTP	Precision Time Protocol, IEEE 1588 (cf. NTP)
RFC	Request for Comments. A publication of the Internet Engineering Task Force (IETF) and the Internet Society
RRSIG	Signature for a DNSSEC-secured record set. A DNS record type
RSA	An algorithm of a practicable public-key cryptosystem (the initials are those of its developers)
SOA	Start of (a zone of) authority record. A DNS record type
SSH	Secure SHell. A cryptographic network protocol for secure network services
SSHFP	SSH Public Key Fingerprint. A DNS record type
TLD	Top Level Domain
UDP	User Datagram Protocol, a network communications method

Complete BPDs available at http://services.geant.net/cbp/Pages/Home.aspx

campus-bp-announcements@terena.org