Advanced DNSSEC

There are several advanced DNSSEC configuration options you can use if you enable DNSSEC for your managed zones. These range from different signing algorithms and denial of existence to the ability to use record types that require or recommend DNSSEC for their use; they are all described below.

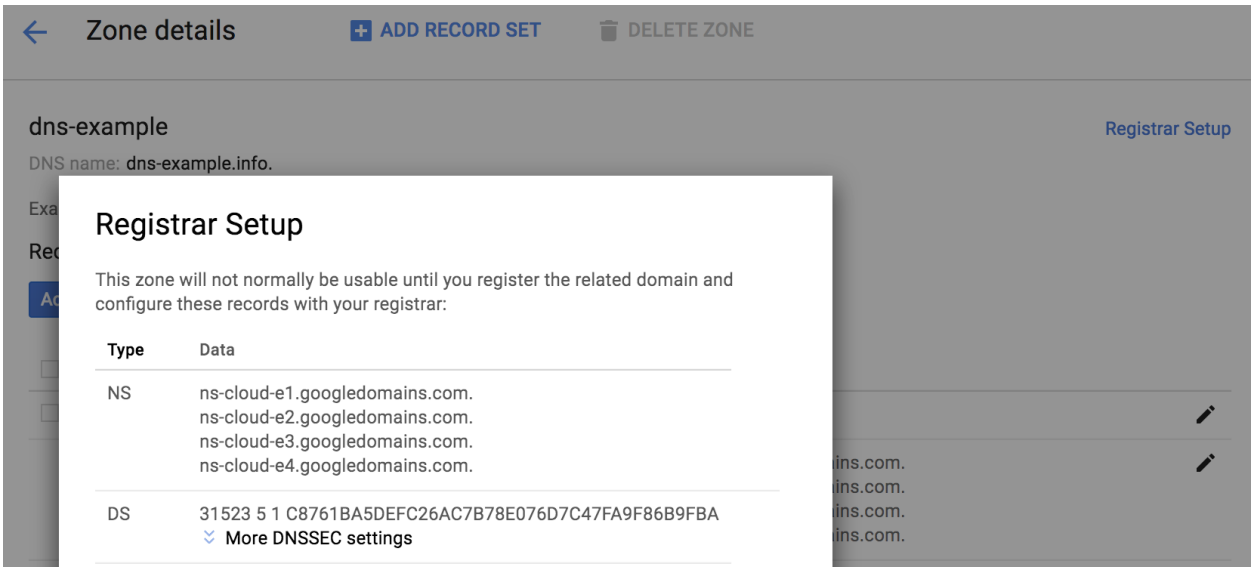
Delegating DNSSEC-signed subdomains

You can enable DNSSEC for delegated subdomains once you have enabled DNSSEC for your primary domain. To enable DNSSEC for delegated subdomains, first create a DS record within a Cloud DNS zone. You also must create one or more NS records. For instructions on how to create NS records, see Adding or removing a record (/dns/records#adding_or_removing_a_record).

Consolegcloud (#gcloud)

To create DS records for delegated subdomains, you must obtain the DS record for the zone. If the delegated zone is also hosted in Cloud DNS, you can get the DS record from the Cloud Console.

1. Navigate to the managed zone.
2. To see the DS record for the zone, click the `Registrar Setup` link in the upper right corner of the `Zone details` page.
3. After noting the DS record information, continue the procedure in step 2 of the `gcloud` commands tab.



Advanced signing options

When enabling DNSSEC for a managed zone, or creating a managed zone with DNSSEC, you can select the DNSSEC signing algorithms and the denial-of-existence type.

You can change DNSSEC settings (for example, to the algorithm used to crytopgraphically sign records),for a managed zone *before* enabling DNSSEC. If DNSSEC is already enabled for a managed zone, to make changes, first disable DNSSEC, make the required changes, and then re-enable DNSSEC using this command (replace ***example_zone*** with the name of a DNS zone in your project):

```
d dns managed-zones update example_zone --dnssec-state on
```

For details, see Enabling DNSSEC for managed zones (/dns/docs/dnssec-config#enabling).

f you want your domain to resolve while DNSSEC is disabled, you must first deactivate DNSSEC for your zone omain registrar to ensure that DNSSEC-validating resolvers can still resolve names in the zone. For details, see ng DNSSEC for managed zones (/dns/docs/dnssec-config#disabling).

The following command enables DNSSEC with 256-bit ECDSA and NSEC for the smallest DNSSEC-signed response packets possible using Cloud DNS. Replace ***example_zone*** with the

name of a DNS zone in your project:

```
d dns managed-zones update example_zone \
nssec-state on \
sk-algorithm ECDSAP256SHA256 --ksk-key-length 256 \
sk-algorithm ECDSAP256SHA256 --zsk-key-length 256 \
enial-of-existence NSEC
```

If you provide any KSK or ZSK algorithms or key lengths, you must provide *all of them* (`--ksk-algorithm --zsk-algorithm --ksk-key-length --zsk-key-length`) and their arguments in the command. You can specify denial-of-existence as NSEC or NSEC3 independent of algorithms.

The supported algorithm options and arguments are listed in the following table showing **defaults** as well as *weak values* only available on request:

Algorithm	KSK lengths	ZSK lengths	Comments
RSASHA1	*1024*, *2048*	*1024*, *2048*	SHA-1 considered weak
RSASHA256	*1024*, **2048**	**1024**, 2048	
RSASHA512	*1024*, 2048	1024, 2048	Not widely supported
ECDSAP256SHA256	256	256	May not be supported
ECDSAP384SHA384	384	384	Not widely supported

Do not use **RSASHA1** unless you need it for compatibility reasons; there is no security advantage to using it with larger key lengths.

The security and performance differences between **RSASHA256** and **RSASHA512** are minimal and signed response sizes are identical. Key lengths do matter: longer keys are slower and responses are larger (see analyses of response size for the root zone (https://indico.dns-oarc.net/event/21/contribution/8/material/slides/0.pdf) and TLDs (https://ant.isi.edu/tdns/keysize.html), and an analysis of server-side performance on Windows (https://technet.microsoft.com/en-us/library/dn593667(v=ws.11).aspx)).

Client support for **ECDSA** is limited to fairly recent systems; older clients that cannot validate ECDSA-signed zones consider them unsigned, which may be insecure if you use new record types that rely on DNSSEC (#using). Registrar and registry support for 256-bit ECDSA is common

but not universal; 384-bit ECDSA is only supported by a few registries and even fewer registrars. Using ECDSA *can* be effective if you do not need to support older clients; the signatures are much smaller and faster to compute.

Avoid using different algorithms for the KSK and ZSK in your managed zones; it reduces compatibility and may compromise security. Some DNSSEC-validating resolvers may fail validation for zones with DNSKEY algorithms that are not used to sign all records in the zone, even though RFC 6840 (https://tools.ietf.org/html/rfc6840#section-5.11) says "they MUST NOT insist that all algorithms ... in the DNSKEY RRset work." If this is not a concern (most validating resolvers follow RFC 6840) you can use RSASHA256 for the KSK and ECDSA for the ZSK if your domain registrar or TLD registry do not support ECDSA and you need reduced response sizes.

NSEC3 is the default denial-of-existence type; it offers limited protection against zone walkers trying to discover all the records in your zone. The NSEC3PARAM settings are fixed: NSEC3 "opt-out" is disabled for security, and there is 1 additional hash iteration (for a total of 2) with a 64-bit salt.

NSEC has slightly smaller responses, but no protection against zone walking. Using NSEC can also reduce queries for nonexistent domains; Google Public DNS (https://developers.google.com/speed/public-dns/docs/using) and several other DNSSEC-validating resolvers can synthesize (https://tools.ietf.org/html/rfc8198#section-5.1) negative responses from cached NSEC records without querying your Cloud DNS zone.

Using new DNS record types with DNSSEC-secured zones

After your domain has been fully DNSSEC-secured you can start using several new DNS record types that leverage the authenticity and integrity guarantees of DNSSEC-signed zones to enhance the security of other services.

SSHFP

SSHFP records contain a fingerprint of an SSH server's public key, and can be used by SSH client applications to validate the SSH servers. SSH clients usually require user interaction to confirm the server's public key on the first connection and generate warnings if the key is changed. An SSH client configured to use SSHFP always uses the key in a server's SSHFP record for that server; only keys for servers without an SSHFP record are saved for reuse.

The SSHFP record format is: algorithm number, fingerprint type number, and server key fingerprint. The algorithm numbers for SSH are

1. RSA
2. DSA
3. ECDSA
4. ED25519

Fingerprint types are

1. SHA-1 (*deprecated, only for compatibility*)
2. SHA-256

n: Improper use of SSHFP records can have serious security consequences; follow these rules to avoid creatin
y vulnerabilities:

Do not create SSHFP records in a zone that is not DNSSEC-secured.

Never configure SSH clients to use SSHFP for a domain that is not DNSSEC-secured.

Never configure SSH clients to use SSHFP unless they validate DNSSEC or use a validating resolver such as Go
Public DNS (https://developers.google.com/speed/public-dns/).

llowing these rules could allow adversaries to create spoofed SSHFP records for your servers in order to
onate them, making SSH connections to them insecure and vulnerable to attacks.

StackExchange has suggestions for creating SSHFP (http://unix.stackexchange.com/questions/121880) and there are tools for to generate them by scanning servers, using existing known host files (https://github.com/rodecker/sshfp/blob/master/README.md), or configuration management (http://jpmens.net/2012/02/01/on-collecting-ssh-host-keys-for-sshfp-dns-records/). These notes (https://confluence.clazzes.org/display/KH/SSHFP+records%3A+DNS+providing+public+ssh+host+keys) have even more examples and links.

TLSA and DANE

TLSA records contain information that can be used to validate X.509 certificates (such as those used by HTTPS) without depending on their being signed by one of a pre-configured set of

certificate authorities (CAs). This allows you to set up your own CAs and generate certificates for HTTPS, as well as permitting validation of certificates for protocols like SMTP where there is typically no application support for pre-configured trusted CAs.

ın: Do not create TLSA records in a zone that is not DNSSEC-secured. Most applications that use TLSA records ɛ DNSSEC validation, so creating such records has no effect; but it may create a vulnerability in misconfigured ations.

DANE (Domain Authentication of Named Entities) as specified in RFC 6698 (https://tools.ietf.org/html/rfc6698) and related RFCs, uses TLSA records in specific ways to secure many protocols and applications. The IETF Journal and Internet Society have a useful introductory article (https://www.ietfjournal.org/dane-taking-tls-authentication-to-the-next-level-using-dnssec/) and DANE overview (http://www.internetsociety.org/deploy360/resources/dane/).

HTTPS

DANE allows you to securely configure HTTPS servers using certificates generated with your own Certificate Authority infrastructure based on OpenSSL (https://deliciousbrains.com/ssl-certificate-authority-for-local-https-development/) or CloudFlare's CFSSL (https://blog.cloudflare.com/how-to-build-your-own-public-key-infrastructure/).

SIDNLabs' DANE validator for HTTPS servers (https://check.sidnlabs.nl/dane/) is helpful for testing a DANE deployment for HTTPS.

SMTP (E-mail) TLS certificate verification

The SMTP e-mail protocol is particularly vulnerable to downgrade attacks that disable encryption (http://arstechnica.com/security/2015/10/dont-count-on-starttls-to-automatically-encrypt-your-sensitive-e-mails/), and DANE provides a way to prevent these. The Internet Society has a two (http://www.internetsociety.org/deploy360/blog/2016/01/lets-encrypt-certificates-for-mail-servers-and-dane-part-1-of-2/) part (http://www.internetsociety.org/deploy360/blog/2016/03/lets-encrypt-certificates-for-mail-servers-and-dane-part-2-of-2/) tutorial on using DANE for SMTP with the free and automated certificates available from Let's

Encrypt (https://letsencrypt.org/). (Be sure to heed the advice (https://community.letsencrypt.org/t/please-avoid-3-0-1-and-3-0-2-dane-tlsa-records-with-le-certificates/7022) to avoid using certain TLSA record formats with Let's Encrypt certificates.)

There is also excellent advice (and a DANE domain validator for HTTPS and SMTP) at https://dane.sys4.de/common_mistakes (https://dane.sys4.de/common_mistakes). The "Test your email" validator (https://en.internet.nl/) also checks DANE.

XMPP (Jabber chat) TLS certificate verification

XMPP (and other services that use SRV records) can also take advantage of DANE. An XMPP example (https://op-co.de/blog/posts/yax_im_dnssec/) uses DANE-SRV configuration as specified in RFC 7673 (https://tools.ietf.org/html/rfc7673).

TXT (OpenPGP / GnuPG) public key association

TXT records can be used without DNSSEC, but DNSSEC-signed TXT records provide greater confidence in their authenticity. This is important for DNS-based publication of OpenPGP (GnuPG) public keys, which are not signed by well-known parties like X.509 certificate authorities. If *Alice* publishes (in the DNSSEC-signed `an.example` zone) a TXT record like:

```
._pka 86400 IN TXT "v=pka1;fpr=083382bac0059be3d6544c8b0dcf16f482a6;uri=https://an.e
```

and hosts a copy of the ASCII-armored public key at the given URI, *Bob* can look up an OpenPGP key for `alice@an.example` with reasonable security (this is not a replacement for web-of-trust validation, but makes OpenPGP encryption possible with previously unknown parties).

You can use these instructions (https://github.com/mattrude/pgpkeyserver/blob/master/doc/public-key-association.md) to generate these "PKA Version 1" TXT records (as they are called in this overview of keys in DNS (https://incenp.org/notes/2015/keys-in-dns.html)). Another how-to (http://gushi.org/make-dns-cert/HOWTO.html) also explains how to create OpenPGP CERT records (but neither CERT nor OPENPGPKEY records are supported by Cloud DNS).

If you have registered your OpenPGP key at Keybase.io (https://keybase.io), you don't need to host the key on your own server and can simply use a URL like

`https://keybase.io/KEYBASE_USERNAME/key.asc` (replacing `KEYBASE_USERNAME` with your Keybase.io username). If you have uploaded your OpenPGP key to a keyserver, you can also use an hkp: URI for that keyserver, such as `hkp://subkeys.pgp.net` or `hkp://pgp.mit.edu`, although users behind firewalls blocking port 11371 may not be able to reach it (some keyservers provide port 80 HTTP URLs.

TXT (SPF, DKIM, and DMARC)

There are three other kinds ot TXT records that can be used to secure your e-mail services and prevent spammers and scammers from sending e-mail that appears to come from your domain (even though it doesn't).

SPF (https://support.google.com/a/answer/33786)

> Specifies the SMTP servers (by domain name or IP address) that may send e-mail for a domain.

DKIM (https://support.google.com/a/answer/174124)

> Publishes a set of public keys used to verify e-mail is sent from a domain, and protect messages from being modified in transit.

DMARC (https://support.google.com/a/answer/2466580)

> Specifies domain policies and reporting for SPF and DKIM validation and error reporting.

You can use the "Test your email" validator (https://en.internet.nl) to verify that your domain is properly configured to use all three of these record types. At the Common Mistakes FAQ (http://www.open-spf.org/FAQ/Common_mistakes/) you can find useful advice on configuring SPF records.

Other record set types enhanced by DNSSEC

Besides TXT, there are a few other record set types that benefit from DNSSEC, even though they do not require it.

CAA

Certification Authority Authorization (CAA) record sets allow you to control which public certificate authorities (CAs) can generate TLS or other certificates for your domain. This is most useful (and effective) if you want to make sure that a public CA issuing domain-validated (DV) certificates (like LetsEncrypt.org) does *not* issue certificates for your domain.

A typical CAA record has a simple format like this: `0 issue "best-ca.example"`; that example allows the best-ca.example CA (and no other CA) to issue certificates for names in the domain where the CAA record set is located. If you want to allow multiple CAs to issue certificates, simply create multiple CAA records.

RFC 6844 provides further details on the use of the CAA record set type, and strongly recommends (https://tools.ietf.org/html/rfc6844#section-4.1) the use of DNSSEC.

IPSECKEY

You can also enable opportunistic encryption via IPSEC tunnels by publishing IPSECKEY records (https://tools.ietf.org/html/rfc4025); the StrongSwan (https://wiki.strongswan.org/projects/strongswan/wiki/503) IPSEC VPN implementation has a plugin that uses IPSECKEY records.

In addition to placing IPSECKEY records in the usual forward zones, such as `service.example.com`, RFC 4025 section 1.2 (https://tools.ietf.org/html/rfc4025#section-1.2) requires security gateways to look for IPSECKEY records in the reverse zones `ip6.arpa` and `in-addr.arpa`.

DNSSEC support for reverse zones is less common than for forward zones, and requires an Internet Service Provider that implements DNSSEC themselves, but there are some that can support DNSSEC for reverse zone delegations.

Note that reverse zones for Google Compute Engine VM external IPs do not yet support delegation. See the Google Compute Engine feature request (https://issuetracker.google.com/issues/35904529).

Next steps

- DNS Security (DNSSEC) (/dns/dnssec)
- Managing DNSSEC (/dns/dnssec-config)

- Managing Zones (/dns/zones)
- Cloud DNS Overview (/dns/overview)

Last updated 2020-07-24 UTC.