The ISP Column
A column on various things Internet

June 2021
Geoff Huston

DNSSEC with EdDSA

The world of cryptographic algorithms is one that constantly evolves. In part, this evolution is required by the nature of the process. The practical objective of these algorithms is not to pose a problem that is insoluble, but to pose a problem that is computationally infeasible. For example, a problem that required an enumerative search through as billion candidate answers may have been infeasible to a computer of the 1950's yet would be readily soluble with a processor used in mobile devices today. Most forms of cryptography want to be resistant to current computational capacity, and resistant to what we anticipate to be computational capacity in the coming years. How far into the future you may want to keep the cipher code a secret determines what you think is a suitably strong cryptographic algorithm setting.

Much of the encryption on the Internet these days is provided using the RSA (Rivest-Shamir-Adleman) public key crypto systems, which is also one of the oldest in this area (RSA was first described publicly in 1977 by the mathematicians whose surnames have been used to label the algorithm). RSA is a public-key algorithm whose cryptographic strength is based on the practical difficulty in factoring a number which is the product of two large prime numbers. What would be considered to be suitable strong encryption using the RSA algorithm has changed over the years to use ever-larger key sizes. These days 2,048-bit keys are regarded as the minimum acceptable size for RSA keys. This implies that systems that use RSA need to cope with larger keys and larger digital signatures. This increasing key and signature size can present practical issues in size-sensitive applications, such as signing DNS records in the DNS and passing these signatures over the DNS protocol.

The response to this escalation in key sizes is to look at alternative forms of public-key algorithms which have a higher cryptographic "density". By this I mean that comparable cryptographic "strength" can be obtained with shorter keys than the RSA equivalent. Let's take up this story with Elliptical Curve Cryptography.

ECDSA Cryptography

ECDSA is a digital signature algorithm that is based on a form of cryptography termed Elliptical Curve Cryptography (ECC). This form of cryptography is based on the algebraic structure of elliptic curves over finite fields.

The security of ECC depends on the ability to compute an *elliptic curve point multiplication* and the practical inability to compute the multiplicand given the original and product points. This is phrased as a *discrete logarithm* problem, solving the equation $\boldsymbol{b}^{\boldsymbol{k}} = \boldsymbol{g}$ for an integer $\boldsymbol{k}$ when $\boldsymbol{b}$ and $\boldsymbol{g}$ are members of a finite group. Computing a solution for certain discrete logarithm problems is believed to be difficult, to the extent that no efficient general method for computing discrete logarithms on conventional computers is known (outside of potential approaches using quantum computing of course). The size of the elliptic curve determines the difficulty of the problem.

The major attraction of ECDSA is not necessarily in terms of any claims of superior robustness of the algorithm as compared to RSA, but in the observation that Elliptic Curve Cryptography allows for comparably difficult problems to be represented by considerably shorter key lengths. If the length of the keys being used is a problem, then maybe ECC is a possible solution.

> "Current estimates are that ECDSA with curve P-256 has an approximate equivalent strength to RSA with 3072-bit keys. Using ECDSA with curve P-256 in DNSSEC has some advantages and disadvantages relative to using RSA with SHA-256 and with 3072-bit keys. ECDSA keys are much shorter than RSA keys; at this size, the difference is 256 versus 3072 bits. Similarly, ECDSA signatures are much shorter than RSA signatures. This is relevant because DNSSEC stores and transmits both keys and signatures."
>
> RFC 6605, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", P. Hoffman, W.C.A. Wijngaards, April 2012

We are probably justified in being concerned over ever-expanding key sizes in RSA, and the associated implications of the consequent forced use of UDP fragments for the DNS when packing those longer key values into DNSSEC-signed responses. As already noted, if UDP fragmentation in the DNS is unpalatable, then TCP for the DNS may not be much better, given that we have no clear idea of the scalability issues in replacing the stateless datagram transaction model of the DNS with that of a session state associated with each and every DNS query. The combination of these factors makes the shorter key sizes in ECDSA an attractive cryptographic algorithm for use in DNSSEC.

ECDSA has a background of patents and IPR claims, particularly, but not entirely, associated with the entity Certicom, and for some time this IPR confusion was considered sufficient reason for many distributions of crypto libraries not to include ECDSA support (http://en.wikipedia.org/wiki/ECC_patents). OpenSSL, the most widely adopted open crypto library, added ECDSA from version 0.9.8 in 2005, but a number of software distributions took some further time to make the decision that it was appropriate to include ECDSA support (such as Red Hat Fedora, where the distribution's inclusion of ECDSA support was apparently delayed until late 2013, probably due to these IPR concerns: https://bugzilla.redhat.com/show_bug.cgi?id=319901)

Taking all this into account, it's not surprising that in the past folk have been cautious with their approach to ECDSA, both to use it as a signing algorithm and to support its use in various crypto validation libraries.

But if neither RSA nor ECDSA are to your taste, then are there any viable alternatives to these two algorithms?

EdDSA Algorithms

The Edwards-curve Digital Signature Algorithm (EdDSA is a more recent offering (first published in 2011) which is a member of the family of elliptic-curve cryptographic algorithms, designed to be faster than other algorithms, unencumbered by IPR disputes and available in public domain source form.

This algorithm uses a "twisted Edwards curve".

> The normal form of elliptic curves that Harold Edwards studied in 2007 was $x^2 + y^2 = c^2 + c^2x^2y^2$. The twisted transform of such curves results from the relationship $ax^2 + y^2 = 1 + dx^2y^2$. These curves can be used to derive a digital signature algorithm for use in public key cryptography, described in RFC 8032.
>
> Ed25519 uses an instance of this curve where $a = -1$ and $d = -121665/121666$ and is mapped into a prime field p where $p = 2^{255} - 19$. This produces the relationship: $-x^2 + y^2 = 1 - (121665/121666)\, x^2y^2 \pmod{2^{255} - 19}$

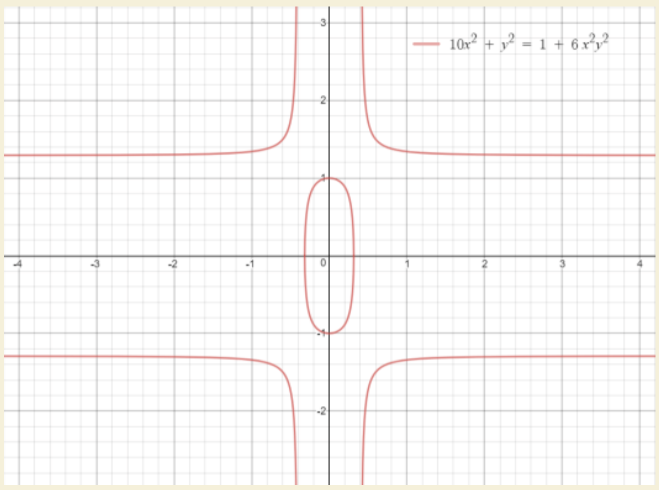

Figure 1 - Twisted Edwards curve for a=10, d=6

If EdDSA algorithms give an equivalent level of cryptographic strength to other elliptical curve algorithms and are said to be faster to compute, then should we all be using EdDSA for our crypto applications? Indeed, do we support this family of algorithms today? Or, to pose the question in a different manner, if I generated a digital signature using an EdDSA crypto algorithm, how many users on the Internet are using validating DNS services that can validate such a signature?

Edwards curve cryptography is a more recent algorithm than ECDSA, with the relevant standard specification for use in DNSSEC, RFC8080, published in February 2017. Given that it takes some time for these algorithms to be accepted as robust, it might be that support for Edwards curve cryptography is not so widespread as the ECDSA or RSA algorithms. Support in OpenSSL for Ed25519 and ED448 was only released in late 2018. On the other hand, there are only a few crypto libraries in use and there are constant reminders to keep the local copies of these libraries up to date because of the risk of exploitation of known vulnerabilities in older versions of these libraries. If we are all keeping such software libraries up to date then support for Edwards curve crypto should already be as widespread as Elliptic curve cyrpto.

The question here is: What's happening with deployment of tool libraries that support these recent crypto algorithms, and EdDSA in particular?

That is what I would like to look into here, using DNSSEC as the service substrate.

EdDSA Measurements

Let's look at a comparison of ECDSA P-256 and Ed25519 in a little more detail.

Firstly, let's look at computational performance to sign a zone file. Here I'll use the OpenSSL 1.1.1k libraries on a FreeBSD 12.2 host with the DNSSEC toolset supplied with Bind 9.16.16. The absolute numbers for elapsed time are not that important here, but what is important is the elapsed time to perform a DNSSEC function across each of the three crypto algorithms being used. Here we use NSEC signing rather than NSEC3 across a zone with 500,000 entries.

Algorithm	Signing Time	Signed Zone Size
RSA SHA256 with 2,048-bit keys	3,200 seconds	800Kbytes
ECDSA P-265	450 seconds	350Kbytes
Ed25519	810 seconds	350Kbytes

Clearly RSA gets a whole lot slower with larger key sizes. It's also the case that Ed25519 takes roughly twice the time to sign a zone as ECDSA P-256. If you are after high(er) speed and smaller record sizes when signing a zone, then ECDSA P-256 is looking pretty good from this perspective.

What about validation performance across these algorithms?

Algorithm	**Signing Time**
Unsigned	872 seconds
RSA	878 seconds
ECDSA	884 seconds
Ed25519	878 seconds

Here we performed a query for 50,000 domain names in zones signed by each of the crypto algorithms using a local validating resolver. In this case it appears that the bulk of the elapsed time lies in the DNS transaction to retrieve the resource records and generate a result (872 seconds, or 17.44ms per resolved name without any form of DNSSEC validation. ECDSA validation adds a further 0.24 ms per signed name, while both RSA (with 2048-bit keys) and Ed25519 add a further 0.12 ms per signed name, or one half the additional average validation time.

As well as these questions of the performance of implementations of these crypto algorithms there is also the question of the extent to which the algorithm is recognized by recursive resolvers. The question posed here is: what proportion of the Internet's end users are using security-aware DNS resolvers that are capable of handling objects signed using the EdDSA protocol, as compared to the current level of support for the ECDSA protocol?

We've already performed a similar comparison between ECDSA and RSA in 2018 and reported that at the time some 3% of users who could perform validation using RSA were not capable of performing the same operation with ECDSA signatures. At the time, these users were located in South Africa, Lybia, Tunisia and Uzbekistan.

We have switched our default DNSSEC signature algorithm in our measurement of DNSSEC validation to ECDSA P-256 as of 1 December 2020, on the basis that any residual issues with ECDSA support were more than countered by the advantages of smaller DNS responses when carrying ECDSA digital signatures. We observed no discontinuity in the DNSSEC numbers from our measurement on the day of this switch of algorithms.

At APNIC Labs, we've been continuously measuring the extent of deployment of DNSSEC for more than seven years. The measurement is undertaken using an online advertising network to pass the user's browser a very small set of tasks to perform in the background that are phrased as the retrieval of simple URLs of invisible web "blots". The DNS names loaded up in each ad impression are unique, so that DNS caches do not mask out client DNS requests from the authoritative name servers, and the subsequent URL fetch (assuming that the DNS name resolution was successful) is also a uniquely named URL so it will be served from the associated named web server and not from some intermediate web proxy.

The DNSSEC test uses three URLs:

- a *control* URL using an unsigned DNS name,
- a *positive* URL, which uses a DNSSEC-signed DNS name (signed with ECDSA-P256), and
- a *negative* URL that uses a deliberately invalidly-signed DNS name (signed with ECDSA-P256).

A user who exclusively uses DNSSEC validating resolvers will fetch the first two URLs but not the third (as the DNS name for the third cannot be successfully resolved by DNSSEC-validating resolvers, due to its corrupted digital signature).

The *negative* test exposes an interesting side effect of DNS name resolution. There is no *DNSSEC signature verification failure* signal in the DNS, and the DNSSEC designers chose to adopt an existing error code to be backward compatible with existing DNS behaviors. The code chosen for DNSSEC validation failure is Response Code 2 (RCODE 2), otherwise known as SERVFAIL. In other DNS scenarios a SERVFAIL response means "the server you have selected is unable to answer you" which client resolvers interpret as a signal to resend the query to another server. Given that the validation failure will happen for all DNSSEC-enabled queries, the client stub resolver should iterate through all configured recursive resolvers while it attempts to resolve the name. If any of the resolvers is not performing DNSSEC validation the client will be able to resolve the name. So only users who exclusively use DNSSEC validating resolvers will fail to resolve this negative DNS name.

The authoritative name servers for these DNS names will see queries for the DNSSEC RRs (in particular, DNSKEY and DS) for the latter two URLs, assuming of course that the DNS name is unique and therefore is not held in any DNS resolver cache).

To test the extent to which Ed25519 is supported we added two further tests to this set, namely:

- a validly signed URL where the terminal zone is signed using Ed25519 (protocol number 15 in the DNSSEC algorithm protocol registry), and
- a second URL where the Ed25519 signature is deliberately corrupted.

What do these security-aware DNS resolvers do when they are confronted with a DNSSEC-signed zone whose signing algorithm is one they don't recognize? RFC 4035 provides the answer this this question:

If the resolver does not support any of the algorithms listed in an authenticated DS RRset, then the resolver will not be able to verify the authentication path to the child zone. In this case, the resolver SHOULD treat the child zone as if it were unsigned.

RFC4035, "Protocol Modifications for the DNS Security Extensions", R. Arends, et al, March 2005.

A DNSSEC-validating DNS resolver that does not recognize the Ed25519 algorithm should still fetch the DS record of the parent zone but will then complete the resolution function as if the name was unsigned and it will return the resolution response.

We should expect to see a user who uses such DNS resolvers that do not support the Ed25519 algorithm to fetch the web objects for both of these URLs.

An Ed25519-aware DNSSEC-validating resolver should fetch both the DS records of the parent zone and the DNSKEY record of the zone. As the resolver(s) will return SERVFAIL for the invalidly signed name, only the validly signed name will result in a web fetch of the corresponding URL (assuming all the resolvers that are in the resolution set that are accessed by this user DNS query all support Ed25519).

Ed25519 Validation Results

The results shown here reflect a measurement undertaken over 40 days spanning May 2021. The first is the daily measure of users who use resolvers that will validate DNS responses that are signed with Ed25519, as compared with the daily measure of users who are validating responses using ECDSA P-256 (Figure 2).

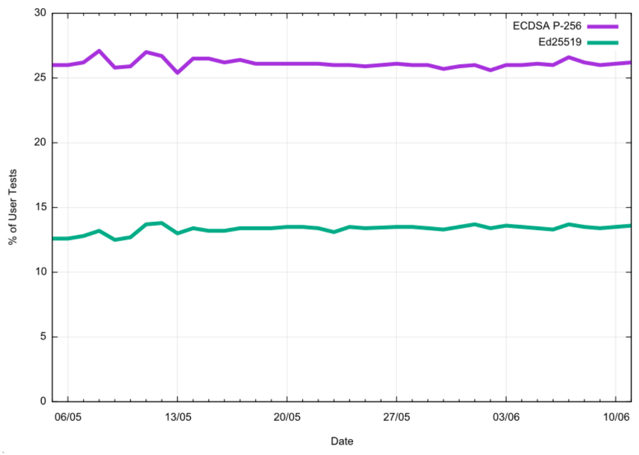

Figure 2 – Support for ECDSA P-256 and ED25519 in DNS resolvers, measured by user count

We can also look at the ratio between these two measurements. This is shown in Figure 3.

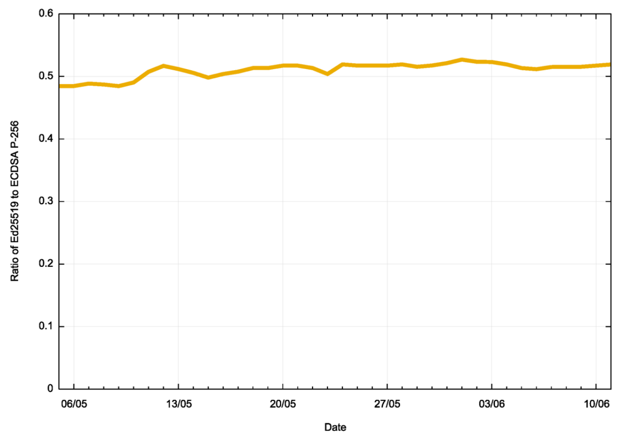

Figure 3 – Ratio of support for ECDSA P-256 and ED25519 in DNS resolvers, measured by user count

This is an unexpected result, indicating that slightly less than one half of all users who use DNS recursive resolvers that perform DNSSEC validation using ECDSA P-256 also treat ED25519 digital signatures as "unknown.

It is often the case that Internet-wide values hide a wealth of detail, and there are very few cases where a global measurement is faithfully repeated in every region and in every network. There is normally a high level of diversity when we look in detail into these various environments, and this area of support for Ed25519 is no exception.

We've generated a world map with a colour scheme that is based on the level of support for Ed25519 in the resolvers used by the various national user populations (Figure 4). The areas where there is a high use rate are coloured green, and red is a low use rate. For high use rate regions, it appears that there is a mixture of national environments where intensive use is made of open DNS resolvers services that support DNSSEC validation using Ed25519, such as Google's 8.8.8.8 service, and environments where there is DNSSEC validation support for Ed25519 in the IPSs' DNS resolver environment.

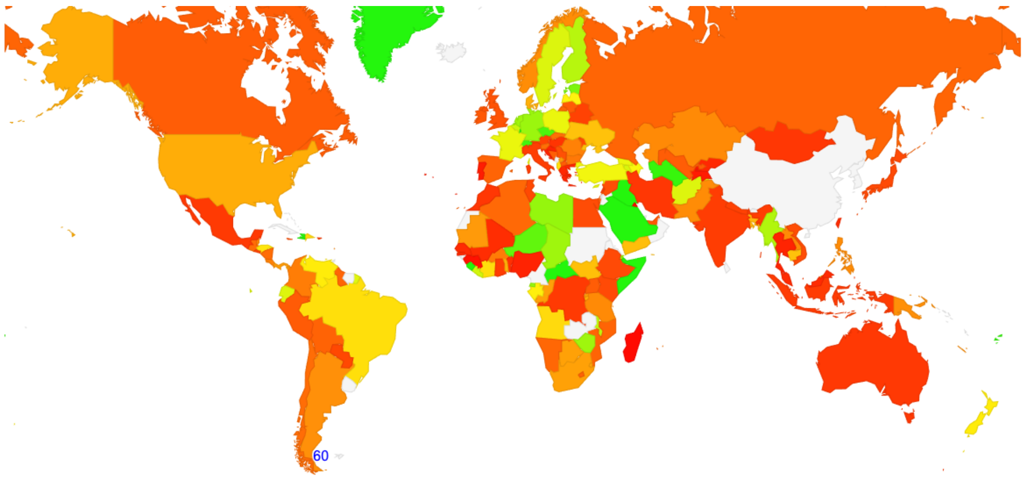

Figure 4 – National Levels of Support for Ed25519 DNSSEC validation

Let's break this down and look at the national economies where these users who use non-ED25519-aware DNS resolvers are located. The "ratio" figure provided here is the ration between the proportion of ECDSA-aware tests and Ed25519 aware tests. Low numbers indicate that there is validation occurring but not with Ed25519 signed records, and high numbers indicate matching support for both crypto algorithms.

CC Code	Ratio	Count	Name
CW	0.00	1052	Curacao
GL	0.00	2238	Greenland
LC	0.00	815	Saint Lucia
HT	0.00	53304	Haiti
CF	0.00	10979	Central African Republic
SR	0.01	1380	Suriname
PF	0.01	8923	French Polynesia
VI	0.01	1321	United States Virgin Islands
CG	0.01	6228	Congo
SL	0.01	85526	Sierra Leone
AW	0.01	6176	Aruba
SX	0.01	1003	Sint Maarten
PE	0.01	120236	Peru
GA	0.02	21263	Gabon
SD	0.02	34113	Sudan
CI	0.02	207957	Cote d'Ivoire
TD	0.02	40283	Chad
ML	0.02	10046	Mali
GF	0.02	3328	French Guiana
BO	0.03	43381	Bolivia
AO	0.03	86031	Angola
LY	0.03	45452	Libya
DZ	0.03	191160	Algeria
PA	0.04	32860	Panama
CL	0.04	133129	Chile

And here are the 25 CC's where there is the highest level of support for both crypto algorithms:

CC_Code	Ratio	Count	Name
WS	0.99	1733	Samoa
KW	0.97	79551	Kuwait
BT	0.96	21411	Bhutan
BS	0.96	14872	Bahamas

KN	0.95	2154	Saint Kitts and Nevis
YT	0.95	2767	Mayotte
IS	0.94	17896	Iceland
LI	0.93	1452	Liechtenstein
MN	0.93	32019	Mongolia
TJ	0.92	49355	Tajikistan
BJ	0.92	89362	Benin
PT	0.91	179247	Portugal
KI	0.91	851	Kiribati
BH	0.91	18526	Bahrain
BB	0.90	5386	Barbados
TO	0.89	1379	Tonga
MA	0.89	886140	Morocco
IN	0.89	19659273	India
IR	0.86	632456	Iran
FO	0.84	2782	Faeroe Islands
GG	0.84	1625	Guernsey
MG	0.83	17317	Madagascar
TL	0.83	8855	Timor-Leste
MU	0.82	37108	Mauritius
BF	0.81	190642	Burkina Faso

Now let's break this data further into ASNs. The following is the list of the ASes , ranked by the number of users that were tested, where we observe a relatively high level of support for DNSSEC validation using ECDSA P-256, but a low level of support for Ed25519.

AS	**CC**	**Count**	**ECDSA**	**Ratio**	**Ed25519**	**Ratio**	**AS Name**
55836	IN	22998060	21848577	0.95	899647	0.04	Reliance Jio, India
7922	US	4070748	3975062	0.98	251901	0.06	COMCAST, USA
36903	MA	2070044	1677701	0.81	148415	0.07	MT-MPLS, Morocco
36992	EG	885949	393591	0.44	64321	0.07	ETISALAT-MISR, Egypt
45245	BD	851055	670971	0.79	22466	0.03	BANGLALINK, Bangladesh
4818	MY	657060	610952	0.93	9234	0.01	DIGIIX, Malaysia
3243	PT	614634	605322	0.98	11220	0.02	MEO-RESIDENCIAL, Portugal
30689	JM	604787	370476	0.61	48738	0.08	FLOW-NET, Jamacia
45727	ID	599026	281679	0.47	27017	0.05	Hutchison, Indonesia
35819	SA	524105	471071	0.90	13819	0.03	Etihad Etisalat, Saudi Arabia
25144	BA	416270	174427	0.42	38926	0.09	TELEKOM-SRPSKE, Bosnia
43766	SA	330041	309512	0.94	10440	0.03	MTC-KSA, Saudi Arabia
8359	RU	281142	273762	0.97	4561	0.02	MTS, Russia
23889	MU	272176	267050	0.98	7127	0.03	Mauritius Telecom, Mauritius
17882	MN	255505	203365	0.80	24229	0.09	MCS, Mongolia
1241	GR	225024	221091	0.98	6626	0.03	Forthnet, Greece
26615	BR	194565	92371	0.47	18433	0.09	TIM, Brazil
37133	TZ	185453	182966	0.99	1529	0.01	AIRTEL, Tanzania
47589	KW	179303	152935	0.85	1207	0.01	KTC, Kuwait
4804	AU	154869	123355	0.80	2380	0.02	Microplex, Australia
6871	GB	133194	112411	0.84	3941	0.03	PLUSNET, UK
9231	HK	114751	89091	0.78	846	0.01	China Mobile Hong Kong, Hong
39603	PL	114310	112859	0.99	1380	0.01	P4NET, Poland
15146	BS	111989	108602	0.97	2871	0.03	CABLEBAHAMAS, Bahamas
37424	BJ	110672	107193	0.97	933	0.01	Spacetel, Benin
12083	US	110585	106278	0.96	10258	0.09	WOW-INTERNET, USA
34058	UA	109748	108367	0.99	1530	0.01	LIFECELL, Ukraine
44143	RS	102026	100691	0.99	1178	0.01	A1SERBIA, Serbia
16086	FI	99675	98390	0.99	1233	0.01	DNA, Finland
29244	BG	97964	96725	0.99	286	0.00	TELENORBG, Bulgaria

Conclusions

What can we say about using Ed25519 as a crypto algorithm for DNSSEC?

On the positive side the digital signatures are as small as ECDSA P-256, and DNS responses will be far smaller than comparable responses using RSA with larger keys. As a rule in the DNS, smaller is better and keeping all responses well under the packet fragmentation size is a definite positive. Ed25519, like ECDSA P-256 is a compact crypto algorithm, and it certainly assists in keeping packet sizes smaller.

On the negative side a significant proportion of users are located behind recursive resolvers that don't recognize the Ed25519 crypto algorithm and treat the DNS response as unsigned. These same resolvers recognize ECDSA P-256, so the problem of lack of validation support is certainly one associated with Ed25519.

Is Ed25519 ready for use?

In my view, this data is telling us "No!" If you want to take advantage of the smaller signature sizes offered by these curve-based crypto algorithms, then ECDSA P-256 appears to offer similar cryptographic strength with the same key sizes as Ed25519, but with a far more widespread support base for validation.

This picture of the relative low levels of integration of support for Ed25519 in DNSSEC-aware validating resolvers will probably change in the coming months. There are evidently ISP DNS resolution products, such as Akamai's Cacheserve, that have introduced support for this particular algorithm only recently, and it may take some time for the various ISP customers of this service to roll out the relevant upgrades across their DNS infrastructure.

However, the bottom line at present is that if you are looking for a more compact DNSSEC crypto algorithm to sign your zone, because you would prefer to avoid the DNS UDP size issues associated with key rolls using RSA with 2,048-bit keys, then ECDSA P-256 would be a better choice than Ed25519, if only because of the broader level of support across the DNS resolution landscape today.

Disclaimer

Author

Geoff Huston B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region.

www.potaroo.net