The ISP Column
A monthly column on things Internet

October 2014

Geoff Huston
George Michaelson

ECDSA and DNSSEC

Yes, that's a cryptic topic, even for an article that addresses matters of the use of cryptographic algorithms, so congratulations for getting even this far! This is a report of a an experiment conducted in September and October 2014 by the authors to measure the extent to which deployed DNSSEC-validating resolvers fully support the use of the Elliptic Curve Digital Signature Algorithm (ECDSA) with curve P-256.

Why "ECDSA"?

ECDSA is an algorithm used to support the use of public/private key pairs as a means of encrypting and decrypting information.

What is ECDSA?

Briefly, ECDSA is a digital signing algorithm that is based on a form of cryptography termed "Elliptical Curve Cryptography". 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 a point multiplication and the inability to compute the multiplicand given the original and product points. This is phrased as a discrete logarithm problem, solving the equation $b^k = g$ for an integer k when b and 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. The size of the elliptic curve determines the difficulty of the problem.

This approach is in contrast to the widely used RSA algorithm, that uses exponentiation modulo a product of two very large primes, to encrypt and decrypt. The security of RSA is based on a somewhat different assumption of difficulty, namely the assumption of the difficulty of prime factoring large integers, a problem for which there is no known efficient general technique.

The major attraction of ECDSA is not necessarily in terms of any claims of superior robustness of the algorithm as compared to RSA approaches, 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.

As pointed out in RFC6605:

> 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.
>
> RFC6605, "Elliptic Curve Digital Signature Algorithm (DSA) for DNSSEC", P. Hoffman, W.C.A. Wijngaards, April 2012

We are 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. If UDP fragmentation in the DNS is unpalatable, then TCP for the DNS probably not much better, given that the problem of middleware destruction of UDP fragments is matched by a similar problem of middleware destruction of attempts to ship TCP traffic to and from port 53. The combination of these factors make the shorter key values in ECDSA a pretty attractive algorithm for use in DNSSEC.

If it's so attractive, then why aren't we all using ECDSA then?

Well, there's one very relevant question that should be answered before you all head off to use ECDSA to sign your DNS zones. Is the ECDSA algorithm as widely supported by DNSSEC-Validating resolvers as the RSA algorithms?

There are reasons why we should ask this question. Elliptical Curve Cryptography is not without its elements of controversy. As Wikipedia explains:

> "In 2013, the New York Times stated that Dual Elliptic Curve Deterministic Random Bit Generation (or Dual_EC_DRBG) had been included as a NIST national standard due to the influence of NSA, which had included a deliberate weakness in the algorithm and the recommended elliptic curve. RSA Security in September 2013 issued an advisory recommending that its customers discontinue using any software based on Dual_EC_DRBG. In the wake of the exposure of Dual_EC_DRBG as "an NSA undercover operation", cryptography experts have also expressed concern over the security of the NIST recommended elliptic curves, suggesting a return to encryption based on non-elliptic-curve groups."
> http://en.wikipedia.org/wiki/Elliptic_curve_cryptography

A similar perspective on Dual_EC_DRBG was the topic of an earlier 2007 essay by Bruce Schneier:

> "If this story leaves you confused, join the club. I don't understand why the NSA was so insistent about including Dual_EC_DRBG in the standard. It makes no sense as a trap door: It's public, and rather obvious. It makes no sense from an engineering perspective: It's too slow for anyone to willingly use it. And it makes no sense from a backwards-compatibility perspective: Swapping one random-number generator for another is easy. My recommendation, if you're in need of a random-number generator, is not to use Dual_EC_DRBG under any circumstances. If you have to use something in SP 800-90, use CTR_DRBG or Hash_DRBG. In the meantime, both NIST and the NSA have some explaining to do."
> https://www.schneier.com/essay-198.html

But let me hasten to note that Dual_EC-DRBG is not ECDSA P-256, and no such suspicions of exposure have been voiced for ECDSA P-256 or its related cousin ECDSA P-384. So while there are perception issues with certain variants of ECC random bit generation, that does not mean that all Elliptical Curve Cryptography is similarly tainted with suspicion.

If we are able to discount such suspicions, and assert that ECDSA p-256 ECDSA P-384 are robust in terms of cryptographic integrity, are there any other problems with the use of ECDSA?

ECDSA has a background of patents and IPR claims, particularly, but not entirely, associated with Certicom, and for some time this was considered sufficient reason for many distributions of crypto libraries not to include ECDSA support (http://en.wikipedia.org/wiki/ECC_patents). OpenSSL 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 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.

The ECDSA Question

The document describing the use of ECDSA as a DNSSEC signing algorithm is RFC6605, published in 2012. This document standardised a profile for use of ECDSA P-256 and ECDSA P-384. Two years later, its probably a good time to ask the question: what proportion of the Internet's end users use DNS resolvers that are capable of handling objects signed using the ECDSA protocol, as compared to the level of support for the RSA protocol?

At APNIC Labs, we've been continuously measuring the extent of deployment of DNSSEC for a couple of years now. (The method we use for this measurement is described at http://www.potaroo.net/ispcol/2013-07/dnssec-google.html, and the results are reported at http://stats.labs.apnic.net/dnssec). 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, and a *negative* URL that uses a deliberately invalidly signed DNS name. A user who exclusively uses DNSSEC validating resolvers will fetch the first two URLs but not the third, and the authoritative name servers will see queries for the DNSSEC RRs (in particular, DNSKEY and DS) for the latter two URLs. This test uses the RSA algorithm.

To test the extent to which ECDSA P-256 is supported we added a fourth test to this set, which is a validly signed URL where the terminal zone is signed using ECDSA P-256 (protocol number 13 in the DNSSEC algorithm protocol registry).

ECDSA Validation Results

What we get back from the ad placement system in this experiment is a set of queries logged at the authoritative name server for the DNS names used in the experiment, and the set of fetches of the named object from the experiment's web servers.

The DNS is one of the best examples I can think of as a meta-stable, non-deterministic, chaotic system that still, surprisingly, manages to operate in a manner that appears to be largely stable and in a highly efficient manner. In the context of analysing the data generated by the experiment's DNS servers, the linkage between the user's DNS resolution subsystem being presented with a DNS name to resolve and the queries that are presented at the experiment's authoritative name servers are often not entirely obvious. The user's local configuration may have multiple resolvers, and when the initial query does not elicit a response within some local timeout, the query will be repeated. Additional resolvers will be

enlisted to assist in the resolution of the name. When a recursive resolver receives such a query it too will have its own repeat timers. And of course there are server farms and other forms of query fanout and both amplify a query and reshuffle a sequence of related queries. The result is that a single user resolution task on an uncached name may result in a large set of queries at the authoritative name server, and the order in which the queries arrive at the authoritative server may not necessarily correspond to any particular ordering of query tasks on the user's original resolver.

However, within all this induced noise there are a number of strong components of signal that points to the actions of a DNSSEC-validating resolver. A distinguishing aspect of such a resolver is that it will query for the zone's keys during the validation phase. One could use the profile of DNSKEY queries as one indicator of the extent to which DNS resolvers are performing DNSSEC validation.

Over 22 days in September and October 2014 the ad network successfully delivered 3,773,420 experiments that performed the embedded set of four tests. Of these experiments, 937,166 queried for the DNSKEY RR of a validly signed (RSA) domain (24.8% of the total). For ECDSA support, the numbers are smaller. Some 629,726 experiments queried for the DNSKEY RR of a validly signed (ECDSA) domain (16.6%).

DNS Query Profile	
Experiments	3,773,420
RSA DNSKEY	937,166
ECDSA DNSKEY	629,726

Table 1: DNS Query Profile

These figures suggest that of the subset of users who use DNSSEC-Validating resolvers, one third of these users are using DNS resolvers that do not perform ECDSA validation.

What do these validating resolvers do when they are confronted with a DNSSEC-signed zone whose signing algorithm is one they don't recognize? One possible action is to regard this as a situation that cannot be validated, and in the same way as a corruption of the key signing data and linkages causes the validating resolver to return SERVFAIL, the use of an unrecognized algorithm would generate a SERVFAIL response rather than the A record. We can gain some further insight into the behavior of DNSSEC-Validating resolvers by combining the DNS query logs with the web server logs, as shown in Table 2.

```
Data collection: 10/9/14 – 4/10/14

552,104 clients who appear to be exclusively using RSA DNSSEC-Validating resolvers

ECC Results:
  Success:       76.45%   361,698  Saw fetch of the DNSSEC RRs and the URL

Fetched the URL but appeared not to validate
  Failure (1)    19.64%   108,411   Did not see query of DNSKEY, but fetched the URL
  Failure (2)     1.47%     8,121   Saw only A queries, but fetched the URL
  Failure (3)     0.84%     4,615   Saw queries with DO set & not set, fetched the URL

Did not fetch the URL
  Failure (4)     1.07%    5,927  Saw query of the DNSSEC RRs, NOT URL
  Failure (5)     0.34%     1,875  Saw query of A, DS, not DNSKEY, NOT URL
  Failure (6)     0.12%       655  Saw only A queries, NOT URL
  Failure (7)     0.08%       436  Saw queries with DO set and not set, NOT URL

  Apparent Fail: 23.55%   130,040
```

Table 2: DNS & Web Query Profile

The data in Table 2 looks only at the behavior of those end users who appear to be exclusively using DNSSEC-Validating resolvers using the RSA algorithm. These clients used resolvers that were observed to validate the two RSA-signed domain names (by virtue of an observed query for the

DNSKEY and DS RRs). The mal-formed signature structure of the second RSA-signed name meant that the resolvers passed back SERVFAIL instead of the address. So the web server saw web fetches from these clients for the first RSA-signed DNS name, but not the second.

It appears that 3/4 of the clients where using RSA-capable and ECDSA-capable validating DNS resolvers, and there were various issues with the resolvers used by the other 1/4 of the clients.

The category "Failure (1)" appears to be significant. These clients do not fetch the zone keys (DNSKEY), so they appear not to be performing DNSSEC validation using ECDSA, but at the same time the clients were passed back an A record from the DNS, allowing them to fetch the web object. It appears that these clients use resolvers that perform DNSSEC validation with the RSA algorithm, but do not perform validation when the signing algorithm used to sign the zone is ECDSA. Are these end-users found in any particular network? The answer is clearly yes, and we noted a number of networks where the majority of end users located within those networks fetched the named object, but failed to perform any DNSSEC validation of the object's DNS name. (The list of AS's that appear to operate DNS resolvers that are not configured with ECDSA is at http://www.potaroo.net/ispcol/2014-10/aslist.txt.)

Is this resolver behavior anomalous? Is a DNSSEC-Validating resolver allowed to switch off validation of a response at it own discretion? What's the "right" thing to do here?

Given that a DNSSEC-validating should withhold the response and return a SERVFAIL indication in other instances when the resolver fails to validate a DNS response, then what should a resolver do when confronted with an algorithm that it does not recognize? RFC4035 provides an 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.

We were very surprised to read this text in the DNSSEC specification. The failover to unsigned occurs without explicit warning to the client who posed the query.

Operators and users of DNSSEC-Validating resolvers need to pay extremely close attention to the setting of the "ad" bit in DNS responses if they want to detect when a DNS response has been downgraded and that validation was not performed. Unsurprisingly, this does not generally occur. The small sample set of operators contacted as part of this experiment to confirm that their resolvers did not support the ECDSA algorithm all expressed surprise that this was the case, and some noted that because their resolvers successfully resolved an ECDSA-signed name, that they could not see that there was a problem at all. Few folk are obsessive with the DNS, and the idea is that the protocol specification should be such that the DNS operates automatically in reasonable ways. We should not need to maintain such a level of manual attention to detail in the huge sea of DNS queries and responses, and we should certainly not require that as a prerequisite to operate a DNS resolver.

This directive for resolver-based downgrading of a signed zone to an unsigned zone is a questionable decision by the DNSSEC developers. Not only does this leave an ECSDSA-signed zone open to the forms of attack that DNSSEC was intended to prevent, but if we take a step

further and use DNSSEC to secure DNS content, as specified by DANE, which essentially provisions secure keys via DNSSEC-protected DNS, then this failure mode will still deliver the key information, but without any validation of the data. The other option open to the design was to return SERVFAIL, which seems somewhat of a harsh measure, but there are mitigations. SERVFAIL does not mean that this name cannot be resolved, but that this particular resolver was unable to resolve the name via a particular server. It is not an authenticated denial of existence, but a message of an inability to validate the response.

We can build on this. It may be wishful thinking, but if a DNSSEC-validating resolver were allowed to return a reason code for SERVFAIL in an EDNS0 field, then it would be perfectly reasonable for the reason code to include a value to signal whether the SERVFAIL was due to validation failure within recognized crypto algorithms or a case of algorithm non-recognition by the resolver (a form of distinction between whether the resolver had a problem (algorithm non-recognition) or whether the response data was mutually inconsistent (validation failure). This additional response information would allows the DNS resolver who posed the query the ability to determine via a local policy setting whether to explicitly downgrade the query and query a second time with the EDNS0 Checking Disabled flag set, or to withhold the response, on the basis that while the response included a digital signature, that signature could not be validated.

Elsewhere we have seen this quiet form of downgrade to an unvalidated response termed a form of "security downgrade" attack. Why is this not the case for DNSSEC?

Is ECDSA Viable for DNSSEC Today?

Returning to the original question that motivated this experiment, is ECDSA a viable crypto algorithm for use in DNSSEC today?

In our opinion these results indicate that, sadly, the answer is "no".

One could use ECDSA in conjunction with an RSA algorithm, and sign the zone with two keys, but its hard to see why this would be a preferred approach over just using an RSA signing algorithm.

This is unfortunate. It's unfortunate because it appears that ECDSA is a robust encryption technology that offers the DNS the possibility of smaller key sizes. This is extremely useful for the DNS as a UDP protocol. If DNS responses require packet fragmentation to pass across the Internet then the DNS will encounter various forms of filtering firewalls and related middleware that is configured to discard all packet fragments, on the grounds that they represent a security vulnerability.

Will ECDSA ever be useful tool for DNS and DNSSEC? As good as ECDSA is in presenting strong crypto in a small number of bits, it is still an alien algorithm for much of today's Internet. So, sadly, I have absolutely no idea how to answer that question as to when it may become genuinely useful for use in DNSSEC.

For those DNS operators who are running DNSSEC-validating resolvers it would be useful if I could refer them to a site that could check their resolver's capabilities with the ECDSA P-256 algorithm.

But even there it's not a case that ECDSA is universally supported: http://dnssec-debugger.verisignlabs.com/test.00001.y.dotnxdomain.net

Sigh!

But, thankfully, other validation checking tools are configured with ECDSA support, such as dnsviz:
http://dnsviz.net/d/test.00001.y.dotnxdomain.net/dnssec/

Some IPR Opinions

It appears that the major issue here is that IPR claims existed over some implementations of ECDSA, inhibiting it's general adoption and use. While there is a body of opinion that certain implementations of ECDSA; NIST FIPS 186-3 do not infringe on these patents, allowing ECDSA to be used in open crypto software libraries on a basis that is comparable with RSA, the lingering IPR concerns have implied that there are still a lot of deployed DNS resolver systems are using platforms where ECDSA is not configured.

The IPR mess that the world of technology has found itself in is a destructive world. Is it really possible that human genes are patentable? Is BRCA1 gene really "owned" by Myriad Genetics, despite the fact that it's an integral part of human genome material? Or if you want to head from the sublime to the obviously ridiculous, can Apple really claim patent rights to rectangular devices with rounded corners?

IPR claims are a sad addition to today's world. In the area of technology, and particularly in the area of open source technology that, incidentally, supports the operation our entire world these days, we are faced with an environment that has been comprehensively undermined by IPR claims. It's not only the claims from known IPR holders, but the potential for the subsequent emergence of IPR holders and the expensive and damaging lawsuits that ensue. We've witnessed the emergence of the patent trolls that buy up existing patents and launch actions against those folk who appear to be impinging on the patent.

Patents were meant to encourage and reward innovators and inventors. They were meant to allow innovation to have an unencumbered period of gestation before allowing open competitive exploitation of the idea. Instead we are witnessing yet another instance of the enclosing of the commons of innovation and ideas. Instead of encouraging innovation and reaping a common economic benefit of greater productivity through technical innovation and invention, our IPR-ridden environment is inexorably strangling it.

Postscript

Google's Public DNS service is used by many users across the Internet. Approximately 1 in every 7 Internet users send their DNS queries via Google's Public DNS. Sometimes these users configure it themselves, and sometimes their ISP just uses the service. Either way a lot of users are handled by this service, and if Google's DNS servers did not validate ECDSA-signed zones, then the results in Table 2 would be far lower.

Google's resolvers certainly look as if they validate the ECDSA-signed DNS responses. Their queries all have the EDNS0 DNSSEC-OK bit set, and they query for both the hash of the zone key (DS) and the Zone keys (DNSKEY) immediately after querying for the A record. From the point of view of the authoritative name server their queries are entirely consistent with DNSSEC validation being performed on the responses. And when we look at the web logs, users who have their queries passed through these resolvers generally are seen to retrieve the URL, so they appear to be getting a response from Google, and not a failure indication.

But if you query Google directly with an ECDSA-signed name, then the response passed back has the "ad" flag clear. In other words, the Google DNS is responding that this is a response from an unsigned zone, and has not been validated.

A review of the figures presented in Table 2, using a filter that assumes that all Google's resolvers used by the Public DNS service are not validating the ECDSA signed zone changes the experiment's outcome, so that the results are inverted.

```
Data collection: 10/9/14 – 4/10/14

552,104 clients who appear to be exclusively using RSA DNSSEC-Validating resolvers

1. Assuming Google's Public DNS validates ECDSA-signed responses:

  ECC Results:
    Success:       76.45%   361,698  Saw fetch of the DNSSEC RRs and the URL

  Appeared not to validate
    Fail:          23.55%   130,040



2. Assuming Google's Public DNS does not validate ECDSA-signed responses:

  ECC Results:
    Success:       24.59%   130,220  Saw fetch of the DNSSEC RRs and the URL

  Appeared not to validate
    Fail:          76.41%   421,884
```

Table 3: DNS & Web Query Profile

The picture for uptake of ECDSA in DNSSEC is more pessimistic than originally thought. It appears that if we assume that Google's Public DNS service does not validate ECDSA-signed zones then some 3/4 of all users who use DNSSEC-validating resolvers do not validate DNS responses when the zone (or its parent) is signed with an ECDSA algorithm. This relatively small base of validators makes ECDSA all the harder to be accepted as a useful compact crypto algorithm for DNSSEC zone signing.

Disclaimer

The above views do not necessarily represent the views or positions of the Asia Pacific Network Information Centre.