ICANN70 | Virtual Community Forum - DNSSEC and Security Workshop (3 of 3)
Wednesday, March 24, 2021 – 12:30 to 14:00 EST

KATHY SCHNITT: Thank you. Hello and welcome to the final session of the DNSSEC and Security Workshop for ICANN70. My name is Kathy and I'm joined by my colleagues Kimberly Carlson and Andrew McConachie and we are the remote participation managers for this session. Please note that this session is being recorded and follows the ICANN expected standards of behavior. During the session, questions or comments will only be read aloud if submitted within the Q&A pod. We will read them aloud during the time set by the chair or moderator of the session. If you would like to ask your question verbally, please raise your hand and when called upon, you will be given permission to unmute your microphone. Kindly unmute your microphone at that time to speak.

This session includes automated real-time transcription. Please note this transcript is not official or authoritative. To view the real-time transcript, click on the closed caption button in the Zoom toolbar. With that, I'm going to hand the floor over to Fred Baker.

FRED BAKER: Hi there. In this session, we're finishing up the DNSSEC and Security Workshop. We're going to have three speakers and four talks. The first will be Ed Lewis talking about ROA deployment in the DNS Core and visualizations of DNSSEC deployment in the DNS Core. What we'll do is we'll take questions between those, between and after. Then, Victor Dukhovni will talk about NSEC 3 and some issues related to it, and Wes Hardaker is going to talk about Cheater's Guide to Algorithm Rolls. So, Ed, can I turn the floor over to you?

EDWARD LEWIS: Sure, and I'll start screen sharing. There. So I have these two talks that are back to back and just to be clear, this is a talk about the ROA deployment in what I call the DNS Core and the later one will be about DNSSEC, so this is the ROA deployment talk first. Again, I have more slides than minutes so I'm going to skip quickly through part of the introduction to get to the heart of this. This talk is going to look at the adoption or the deployment of the Root Origin Attestations to the route advertisements that are put out for name servers in what I call the DNS Core and I'll skip to this slide here. The DNS Core—and I have a cartoon version of it here—basically the upper levels of the namespace. The root zone, the TLDs, and for the purposes of this talk I include the zones that are part of the reverse map that do 1.in-addr.arpa, 2.in-addr.arpa, and the equivalent IPv6 ones which I can't recite without stumbling on myself.

Those are my TLDs for the purpose of this talk. I'm not including things outside of this core like the commercial registrations below that and so on. To be clear about what a ROA is—for those who aren't routing experts—a route origin is a combination of the IP prefix which is advertised to take traffic into and promising through BGP to wind up at a certain ASN that's the last hop of this and the attestation is an X509 certificate which is just a signed way of saying, "I promise, this is what's supposed to happen," basically. It's for the root operator to declare to the world that this is my intention for this prefix. The question I had was, this ROA signing out there, let me take a look at this in the context of a particular application with that application being the DNS. This is sponsored by or came out of someone talking about whether the operators were validating and I thought it's more interesting to me if the operators of DNS servers are assigning these things.

The measurement method here is pretty simple. I take all the zones in what I consider the core, which is the root zone. It's delegations in arpa, it's delegations, so on, for a bit of the tree. I look at all of the name servers involved for these areas, mostly the authoritative name servers for each of these zones in there. I look at each name server and I'm looking for all of these addresses, the IPv4 addresses, IPv6 addresses, whatever address it has. Some have multiple addresses, some only have one. I look then at the route origination information relying on Team Cymru's IP to ASN mapping service. I plug in an IP address, it tells me who's originating a route for it. Rather what route is originated for it and then I look up in the validated by RIPE list of ROA that the route origin attestations and if it says it has one there, that's a yes. If there isn't one there, it's a no, and I just simply do percentages dividing yeses by yes plus nos.

So all of that comes down to a pie chart like this and this was done a few days ago. It shows that roughly a little more than a quarter of the routes involved with the core are covered by a ROA. Which is you might say that's this small number. It's up to interpretation but it's not the majority, obviously. On the right-hand side, I have some ancillary information here. The number or zones I'm counting across is 3,000. That includes sub-TLDs like for example the often cited co.uk for UK. It includes all the number TLDs out there. Down below that TLD's includes we talk about 1,500 or so delegations for the root zone. The reason why I have 1,773 here is it includes the reverse map zones in there too. The number of name servers, around 4,000. Addresses 6 – 7,000. Route origins is 2,000. At first, I was struck by how small those bottom three numbers are. If you had expected each zone to have its own set of name servers you'd want a multiple of 3,000. We don't have that.

Addresses, multiple name servers, we don't quite have that either. It's actually less than 2:1. Route origins shows a lot. Basically, it shows a lot of sharing of what's going on down in the core of the Internet. But so let's look at this ROA coverage. On the left side, you see a chart which shows us a steady upward to the right graph which shows the growth over the last month to month of these measurements. To put that in context, the right-hand side shows a graph which shows instead of going from 22 to 26%, it goes from 0 to 100%, and that considerably flattens the curve and roughly at that rate I calculated roughly we have 10 years before we get to 100% if we just follow that line.

Now, I want to go a little bit deeper to this because there's more to the story. In putting this together there's a lot more things you can learn about the Internet and about how it's put together and I want to know about decision points here. Who was making a decision about whether to deploy a ROA or not and if you're trying to promote that you'd want to ask those people why they haven't done it or basically go to those people and say, "What would it take to get this deployed," Because they may have some really good reasons why it's not been adopted so far. The first thing that routing people like to do is split v4 versus v6. One time it was said that v4 was traditional, v6 was experimental. That's not quite as true anymore but still, there's this idea that v6 is newer than v4. So it's not too surprising to me at first glance that the IPv6 count is a little bit higher but not by much. Very close in the green quadrants here, but I did find something very interesting in the slide here. I should practice using pens here.

The number of TLDs I counted for IPv4 is 1,773. In v6 it's only 1,753 and I went back and thought I had a mistake in my code. It's true that there are 20 TLDs that still don't have IPv6 name servers out there so for those who are promoting v6, it's a place to look for a similar promotion. Another way to look at the split of the ROAs is by the basically a kind of one classification of TLDs. We have the generic TLDs, those are all the TLDs that are not specific to our jurisdiction. The ccTLDs which are traditionally the jurisdictional-based TLDs is another category and then reverse map. I split the reverse map because a lot of the regional Internet registries have been doing a lot of promotion of ROAs in the past couple of years and it's no surprise there that two thirds of their routes are covered. Only about 40% of the ccTLDs are covered and a small amount of the gTLDs are covered.

So obviously we can see some a divergence in the way ROAs are being deployed. Again, I want my decision points and so I look at the layering of registries and I'm going to go to the next slide here to save some time here. I went through a mental exercise, I broke up how a registry is put together. We have the front door on top which is the policy people. We have the database which runs what people are registering the names into. We have below that the DNS which is drawn from the database and DNSSEC is applied there. The DNS server hosting is next level down and then equipment racks, connectivity routing. And if you think about the way—these layers are combined in different ways. Some houses do everything themselves. Some do everything but the routing, some do everything but the connectivity. But there's some that do just the registry front end and some that do only the backend.

I broke this into three parts. Registry Admin, I call that and I don't really go much further with that. I have the DNS zone operators and these are the ones who put together the zone file. They set up the SOA record, for example. Then down here I have server hosting provider. There are some DNS server operators out there that will take anyone's zones and just put them on machines and split them out there. To cheat ahead here, I'm able to group DNS houses by up to 250 TLDs, roughly, and for servers, I can group up to 600 TLDs belong to what I call an autonomous system house. So a DNS house, this is a concept that I've been working with for some time. It's determined by the SOA record, the RNAME field responsible which actually tends to work pretty well in the core.

Operators in the core fill in their RNAME field with something meaningful. It's not necessarily true throughout the protocol. The IANA function also has a DNS root registry and a technical contact field, and I use those two together to create the house. By that, I define what DNS zones are co-managed or co-operated because they tend to move at the same time together. I look at those, I bucket up the TLDs and I notice that I have a couple of large houses. Basically, legacy operators that do hundreds of TLDs but then there are also lots of small ones. I plot this out here, this chart here is—up the Y-axis—is the number of TLDs managed by what's on the [dot.] So this is a large operator, they run a lot of TLDs out there. This runs a lot of TLDs out there. These ones down here are usually doing one or two at a time. Now from left to right, I go from 0 to 100% of ROA deployment, meaning that a lot of these dots up here show large operators that have not rolled out ROAs very much.

It hasn't caught on widely up here. It's scattered across the bottom and the small players are spread out pretty well, and then there's 100% line there's a couple here. Now, I don't really see a whole lot in this but let me also show you what it looked like seven months ago. This is the chart from July 21 when I started taking the data this way. The red circles are the dots that I identify as having not moved and I don't have a better way to represent this right now because I just threw this slide in here but the scale is different. If I go back you see that this one here is above 250. Before that, no one was above 250. There's merger and acquisition activity going on, these are dynamically bucketed things, so things that [tend to] change.

You don't really see a whole lot of movement in the four that are circled, they're pretty much in the same spot. They've moved a little bit but not a whole lot. So the AS house is a little more complex and subjective, where I take the network name from the autonomous system that is homing the server. I look at some shared BGP prefixes, imaginative parsing of network names because there's a lot of variability, what's written in the RIR databases. I've been playing with this a little bit but it's been [steady] for a while, and this shows this chart which looks similar but I started grouping things a bit here. The green dots represent what I call the large operators. These are operators that are legacy, they've been around for a long time. They tend to be all in this low deployment. This one has actually budged off the list here a little bit but most of them are 0% and the red dots here represent the RIRs. Of course, you see the four that are at 100%. The reason why we only see four is the fifth one is actually down there—you can't see it in this chart—and you have an orange one which is a special case of an operator who runs lots of servers for TLDs but does not run a DNS database themself.

But let me get back to what's going on over here. You notice in the RIR pie chart there were only two thirds signed but this shows 100%. That's because if an RIR is running their name servers and they control the routing, they have ROAs. They have some partners that they are making use of who do not do ROAs so that's why that the pie chart seems to conflict with what's here. This chart here I threw in I've been debating on. This is just the small ones. Let me go back here. This is from 0 to 600 TLDs in an autonomous system house. This one is only from 0 to 50 just to highlight what's going on down here in an effort to try to find some kind of pattern, which really, isn't there but there's the fifth RIR showing up there. So overall deployment is sparse.

The large non-RIR hosters have low deployment. The large RIR hosters have high deployment. And large in this case means lots of zones and I already said that the RIRs are counting each of their numbered things [inaudible] map separately that's why it gives them a large count, rather. I do see it's more of a routing thing than a DNS thing in some of these charts here. But let me go onto this. Some people have asked why things are slow here and I think there's an inherent risk of adding security to an in operation system. I think that going slow is a responsible way to go. I have a chart on DNSSEC coming up but it happens time and again when we start bringing security we're always slow with it because if you have something running today that's running well but it's vulnerable, it's more valuable to keep that running and deal with fighting some of the attacks than to risk it falling apart because we didn't set up security correctly.

That's, in general, a philosophical point of view here. There have been concerns/observations that were documented by one of my colleagues in this report down here OCTO14 document. This is a URL to it down below which is an RPKI technical analysis which goes through some of the items observed about the RPKI system and ROAs and concerns that may be keeping it from going much faster. DNSSEC is an example of this. It's post-operational phase security enhancement. It's only gotten to respectable or visible after two decades of deployment, which I find that to be very much expected. We see DNSSEC though has a different adoption pattern. The large operators have deployed DNSSEC in the core specifically, what remains are the single ccTLD operators and I'll get to this in my next talk, actually. I'll talk to you about what I mean by that.

So the wrap-up for this talk here is it's a temperature of the room. 27%, is it acceptable for now? Some people say yes or no. It stirs some ideas of what could we do to promote acceptance. Is it a business case education issue or are there actual gaps that need to be filled? And relying on experience from DNSSEC adoption for the past couple of decades, there are certain advantages to slow adoption, outages will have limited impact. Pioneers are usually the ones quick to address operational problems. That's what actually got DNSSEC through some of the early days when it had problems, people cared to keep it going. There are gaps that DNSSEC [inaudible] and a lot of that was covered in the session a couple hours ago with Steve Crocker's panel. Finally, the value of proposition of what we're doing with security changes over time.

Some time, we may not see this as being very valuable and secure, and suddenly that rise—in fact, that's what happened to DNSSEC and the people have talked about the Dan Kaminsky research paper that was presented 2008, I believe it was which made people aware that the DNS has got to be secured, so that changed public perception over whether or not we should sign and it was a big leap forward in about that time period. With that, I'm done with the first talk and I'll open up to see if Fred, if you want to interrupt for questions now.

FRED BAKER: Well, yeah, let's ask whether anybody has any questions at this point. If so, please raise your hand. And failing that, we'll move along.

EDWARD LEWIS: Okay, I'm going to stop my slide share here.

FRED BAKER: I don't actually see any hands raised, so why don't you go ahead, Ed?

EDWARD LEWIS: Okay. Let me go to the other one. Did I get the right one? No, this is the wrong one. There's some confusion I have with multiple presentations here. This one says it's going to be the DNSSEC one, but it's not. Let me close this one out. Fred, do you see the DNSSEC one?

FRED BAKER: I see a slide that's blue and says DNSSEC Deployment Among TLDs.

EDWARD LEWIS: Okay, good. We're onto the second set so I'll fire away. This is a different topic. This is DNSSEC deployment, this is based on looking at the history of what's happened, again, at the core but this time I'm talking about just the TLD, just the first strings in a domain name because I have data going back almost 10 years on this. I'm going to talk about how DNSSEC deployment has happened over time amongst just the TLDs and the root zone and some of the higher layer zones out there and then a couple of choices that are made here. To give context, let me start with this, when I see a chart that has this shape, this is showing everything that is—and again I'm going to go to a pen here—this is showing all the TLDs I have out there. You'll notice that from late 2013 to 2016 there was about 400 added per year.

This is for those who've been around for a long time was the new gTLD era. This is when we added the people talk about 1,200 zones and that was the rise in that. All of those, the new gTLD program were required to have DNSSEC so it'll talk about DNSSEC adoption. Focusing on this we're just overwhelmed because they all came online signed it from the start. You do notice that there's still a tail off up here. Down below here in the blue, this is ccTLDs. They do grow over time and I'll talk about them separately, they behave more independently. A lot of the gTLDs tend to work in unison because either same backend operator or because there's a contractual obligation for them to behave a certain way so things are controlled there.

By the way, arpa and root don't appear here, they're just too small to be seen. This chart, for example, which shows a smaller chart. This is ccTLDs and I put this up just to give you the shape, it shows the regions. The Asia Pacific area has the biggest number of ccTLDs. Purple is the European area and so on. These are different region by region and this is using the ICANN meetings version of regions. Let me jump into the deployment. So today looking at the pie chart of the current situation, we see that if we look at just the TLDs—and this is counting the 1,504 top-level items out there—91% of them are fully done with DNSSEC. They have everything they need for DNSSEC to run. There are 10 which published signatures and this basically means they don't have a DS record and in a perfect world we'd have 1,503 full ones out there and one down here, this one being the root zone because there is no DNS record for the root zone. But that's not what we want to focus on really.

120 have not signed. 120 TLDs out there that are not DNSSEC signing right now and when I did the chart there was one that showed only keys out there which this is a state that comes and goes. In fact, it's gone

already by today. Whoever had the key in there, they were just coming online and they didn't sign but they had keys out there so that'll go away. As of December, the old gTLDs—being defined as anything that's not a ccTLD—completely done. They're all DNSSEC signed, they all have their DNS records. Then I'll go onto there, ccTLDs are where the action is right now for growth. Most of them are signed, 60% of them are done. About 40% are not yet and there are nine out there that produce signatures that shows keys and signatures but no DS record and some of them have been in that state for a decade.

I've actually tried to talk to some of them about that and ask why and they just didn't feel like they wanted to go that far. Again, if you look at and add numbers up here, these two numbers don't add up to 1,504 because arpa and the root itself are neither—they don't fit these two categories. So ccTLDs trend over time, they are increasing in total and they're also increasingly being signed by DNSSEC, that's what this chart is showing here. Now, besides adoption, one of the bellwethers for how things are being run is to look at the DNSSEC security algorithm. That's the keys that are being used. DNSSEC security algorithm is the proper term for where the key is. It includes cryptography and a hash algorithm. Now the bestest algorithm has changed over time and being a non-cryptographer who's done a lot of protocol development, it was frustrating to me to not know which is the best one at the time.

I like to look at this. Also, a point in the charts coming up here, I wasn't able to quite completely fix this so it was clear, but a TLD may have more than one algorithm at a time and it does that when it just rolls, for example. So, let's look at all the TLDs out there. I'm on actually March

15th and March 16th when I did the chart. Predominantly RSA_SHA256 is in use, almost 90% of the keys out there are that kind. We have other algorithms out there that go down the chart here. It's not always been basically this way but it looks like this order is pretty stable recently. Although ECDSA does charge above RSA-SHA1N which is for NSEC3, at times in some regions, but I won't go that deep in this talk. We have here, this is over time and you can see that we have this growth for that period of time and I'll talk a little bit more about this last year because things got pretty interesting in the last year and that's this part here.

I'm cheating here by this is the last—since the pandemic basically. People were busy, they've made lots of changes here. I used to think that this rise and fall was due to key rollovers and I think that's what it is and I need to go back and check. The one thing that's significant here is this real purple layer of icing here, that represents the elliptic curve ECDSA that's being put out there. I think that was probably this precursor to people switching to that although I'm not sure. The other thing that's happened too is at the end of the last year and especially this year, a real fall in RSA_SHA1 basically. Which is, for those who are not fans of RSA_SHA1 that's good news and that's been really picking up steam in the last part. So I've been surprised on one hand that operators have been configuring in the last year despite everything going on.

But the rate of change, really, we're starting to see something here. Cryptography choices in ccTLDs, this chart shows the prominence of ECDSA. It's number two here although it's number three when I look at all of TLDs and I see here a huge migration away from RSA_SHA1 by ccTLDs. But there's one other thing in this chart I was looking at recently that was interesting, this is a very choppy area. I see a lot of up and down in here, a lot of jagged stuff in this and this is monthly data. Then, here that jaggedness seems to disappear, much smoother. What I've seen is a lot of maturity in some of the operations out there, this is a subjective comment that operators… a lot of this has to do with instability and changes and things either not going well in what was being published and at some point, we seemed to have fixed a lot of that. There was probably a tool that came out that time that had to do with this and I will admit that I did change my way of measuring but it wasn't until about here that I had to rewrite my software.

So it's even the same collector going on for years, the same machines, and so on. I can't think of any reason but I just think the operators are getting more mature in how they deal with these things. This is, again, one more chart. This is repetitive and I won't spend much more time except this one really shows the market share gain of ECDSA in the game of who's signing with what protocol and really shows the shrink of RSA_SHA1 which is actually the yellow and the black. The yellow is for NSEC3, the black was the traditional but they're both the same. The next thing I want to move onto is the number of keys. The number of keys started out as a trivia question, and at one time we were concerned about having too many keys. It had to do with the KSK rollover of 2017 – '18 whether the key set size was too big and so we looked at the number of keys and we saw some interesting things out there.

There was one TLD that had a huge number of keys, fell over. We felt bad because we didn't say anything about it but it turned out it wasn't the keys' fault. But anyway, nevertheless, because I was playing with some tools, this is the average number of keys in a zone going back for 10 years and what's interesting here is it doesn't look interesting at first but you see these little heartbeats, there's actually something in that. Those are key rollovers by some of the big players and so I've been teasing the data a bit to pull that out to see what I can see and so I played a little bit with this. These are rather abstract graphs but each color here represents a different operator. This is bucketed by the DNS Houses that I went back from the previous talk. I look at, over time, some of the things they were doing here and I want to actually skip ahead one more slide.

This shows only from 2018, this makes each day a little bit wider so you begin to see a bit more here but the top one here shows a large operator who regularly changes roughly monthly I believe. I haven't counted them up but every month or so there's a blip here where they change their keys. Another one down here does it looks like every six months and it's a prolonged change where they actually have an extra key out there for a longer period of time than the upper one does. The others here I can't say much about because that's a lot of operators [bumping together,] but these other three, what they show is over time while this case here looks like an operator here did some kind of a transition where they didn't change their overall structure. But this one reduced the number of keys they had by a bit. They took probably one or two keys out.

These heights are multiplied to make them more just dramatic. And this operator also. Now I don't see a lot of evidence of key rollovers popping up in here, either they're too quick or they're not being done but there is a trend in operators to make somewhat significant changes at time to their infrastructure and to reduce the numbers of keys to combat the size of keysets out there apparently and that may be one of the concerns out there. It's interesting to look at these charts because it tells me whether operators are changing the way they think about the protocol. I have a couple other statistics out there and I threw these in a little later on. [NSEC versus NSEC 3,] most people do NSEC3. This is just the choice of NSEC3 versus NSEC in the TLDs. ccTLDs, it's pretty much the same split.

This is over time and the only thing I'll spend here is that you see this, I thought this was a mistake at one point. This actually because I'm doing every month to month, on the first of that month, somebody switched from doing NSEC to NSEC3 so for a while they had both and it happened one more time up here. So operators have occasionally made changes from one to the other but largely it's not, once they make a change and stick with it, and again NSEC3 is so predominantly the choice out there. Now, iterations has become a thing. I went through here and this chart's a little less than clear because I just added it a day or two before I made the slides. If I look at the number of iterations out there and I bucket them into from zero to 10, there's a total of like 82% are in that area. Very small number of iterations.

There's 186 that have an iteration count of 100 and for the most part, they're related to one particular operator I believe. This should actually be 10 of 20 and 38 and so on. These ors mean that there's been changes recently, there's been some attention brought to bear on the number of iterations being something to look at and so a couple TLDs in the last couple days have been prompted to actually bring themselves down, I believe they went down, and ccTLDs, the same here. The good news here from this chart is almost all of the TLDs out there have a low number of iterations with 100 being the highest that's actually seen anywhere and I think that with time that too will come down, but I'm not going to make any promises because I'm not even sure who is doing it, top of my head. Salt length, this is not terribly detailed but the length of the salt that's put in there, it's generally pretty short. There's a 16 byte one out there, there's a 14 byte one out there. There's only one, everything else is either small 0 or very less than 10 bytes long. Usually, it's 4 or 8, frankly.

So the DS Algorithm Hash, I had this as saying a little more exciting than NSEC3. NSEC3 is becoming more interesting so I should strike that first bullet up there right now. NSEC3, or DS Hash, this one, the issue here is that we started out at one time having just a SHA1 as our only option for DS records. Later on, SHA256 was introduced and as a basically training wheels option we let people do both. In the early days it was SHA1 only, then there were those who did both. Nowadays SHA1 really doesn't necessarily have to be there anymore, frankly, and there aren't many left. There's three TLDs that only have SHA1. There's one ccTLD that has SHA1 only. The root zone itself was SHA256 only—or I'm sorry, I take that back because there is no DS record for the root zone, that's a trick question. But we don't have that up at the top anymore.

Everyone has to know SHA256 to run the security in DNSSEC anyway at this point as a minimum. But anyway, we see that most of them usually have 256 or both, but what's interesting here is that—and this to me was unexpected—in the last year along with the changes to the cryptographic algorithms we see that there's been a significant movement away from having both. People are removing the DS record that belongs to SHA1 and they're keeping the SHA256, of course, they need to have that so they shift that. It saves some space, saves stuff that's in databases and it's just better hygiene in many ways for the way the protocol's put together. This was going to be an unexpected part when I put these charts together. There's actually a significant change after being fairly mundane and boring, frankly, as an observer of data. So, questions at this point here. I'm open for questions over this stuff here and I'm interested in suggested visualizations. What's interesting is changing over time. At one point people wanted to see signature durations but that seems to become passé.

Algorithm rollover is now where we are and I think that's reflective of the maturity of the aging of the operators in place. There has been, one of the comments I'll have there is that I've seen cases where a pioneer will come into a TLD and decide to make changes and add things like DNSSEC and all fancy bells and whistles and there's someone who really listens to all of the IETF mailing lists and they read everything and they're up-to-date but they're not necessarily an operator, operator. So when they leave, they leave behind this system which is doing a lot of complex and high tech stuff and when operators are pulled in afterward they are blindsided by some of these bells and whistles and they get caught in the bind sometime and people think the operators have gone down when the fact is operators have gotten better, it's just that the pioneers hadn't left behind a legacy of documentation.

But we're into that era and that's something we need to look at amongst other things involved besides the other gaps that we're seeing out there. With that, I will stop my rambling and open for questions. I think there's a few minutes.

FRED BAKER: Thanks, Ed. There is a question that came up in the chat. Are the ccTLD operators that you have marked as none, are they coming to ICANN meetings?

EDWARD LEWIS: I haven't been following up on that. I haven't, in the last year or so when I've been revamping a lot of my software and of course not traveling I have not been able to go up to that. There's a little bit of reluctance from me to approach some of the ccTLDs frankly because I don't want it to come across as trying to impose a requirement on a ccTLD in the way they operate. There's a legacy of independence there but there is cooperation. In fact, last week I found something going on with one ccTLD and I mentioned it to someone who had mentioned it to someone and I was surprised that within about a few hours they had reconfigured to fix the issue.

It's not like there's a stonewall between us but I haven't made an effort to evangelize and try to go through this. I'm speaking for myself and not for other people in my organization. My Work Group who are trying to promote this, we're trying but it's got to be done in a collegial way as opposed to looking at this as if we're mandating what's going on.

FRED BAKER: Okay. Several comments in the chat about this. So basically, TLD ops maybe should look at DNSSEC as a target and something that it should consider.

EDWARD LEWIS: Yeah, I would say that it's probably time for the protocol to have a, "You're now 25 years old, let's do a checkup on how these specs have been interpreted, how the software's been implemented." Just having a look at the trends of things out there. One thing to keep in mind is a lot of these things I'm poking at operationally have no negative impact, things are working pretty well. It's just that it becomes hard to operate things when you have so many things which are just not lined up perfectly well when you hand off the operations to the next team out there that's coming up next. But we have new people coming in all the time and we want to make it easy for the new operators to handle.

FRED BAKER: Okay, and then hands raised. There was one there, Eric, and I don't know how to pronounce his last name. That Eric did you want to get in? Okay, I'm assuming that he did not since he didn't respond.

EDWARD LEWIS: I saw another hand mention ... I don't see the chat in my screen right now for some reason.

FRED BAKER: Yeah. Okay, so this is…

JACQUES LATOUR: Fred? Jacques here. I've got a couple of positive on TLD ops looking at DNSSEC, so we'll take that on, on our next standing committee meeting.

FRED BAKER: Okay, thank you. Okay, I'm going to take a stab at this name. Erik Østlyngen, you had your hand up and you've now taken it down. Did you want to get in? Okay, so he says, "No, that was an error." With that, I don't see any other hands. Oh, a question from Yoshiro…

EDWARD LEWIS: Yoneya.

FRED BAKER: Yeah. So, did you see that?

EDWARD LEWIS: No. I know the name. I just got out of the screen share so I could look at the chat. He's here.

FRED BAKER: Yeah.

EDWARD LEWIS: Let's see. In the chat, is it in the QA questions? No. I don't see questions, sorry. My version of chat just shows the session beginning.

KATHY SCHNITT: Ed, Yoshiro's asking, have you measured ROA deployment in public DNS resolver providers?

EDWARD LEWIS: No. I have not measured the ROA deployment for public DNS providers. Generally, right now I'm concentrating my work on the authoritative servers that are closer to my work area, frankly, but it's a good thing to do to look at the public DNS resolver providers. The reason why I have been hesitant to actually do work on public DNS providers also is that I have to build a roster of test subjects and right now it'd be very subjective for me to do that. I have actually made a stab at that by looking at some sources for that but that's a good thing to look at. I think that'd be something interesting and if someone wants to take it on as defining a roster of who should be tested as a public DNS resolver, I could run the same code off of that or we could use the same method, frankly. But I think it's a good suggestion it's just the problem there is knowing who to include as a public DNS resolver for the purposes of measurements.

FRED BAKER: Okay, so I don't see any more questions. So let's move onto Viktor.

ULRICH WISSER: Can I ask a question? I had my hand up.

FRED BAKER: Sure, go ahead. Go ahead.

ULRICH WISSER: Okay. Ed, is your code for the rollout stuff available somewhere?

EDWARD LEWIS: It's actually horrible code.

ULRICH WISSER: Everybody says that.

EDWARD LEWIS: Yeah, I'll have to see what I can do. I think some of it may be available to look at it. Let's talk about that offline.

ULRICH WISSER: Okay.

EDWARD LEWIS: Yeah.

FRED BAKER: Okay, so, Viktor.

VIKTOR DUKHOVNI: Hi.

FRED BAKER: Hi there.

VIKTOR DUKHOVNI: You see my slides?

FRED BAKER: Yeah, I see your slides. NSEC3 Iterations, etc.

VIKTOR DUKHOVNI: Right. Okay, so I'll get started. I'm going to be talking about some work that Wes and I are doing, we have an Internet draft with new recommendations on NSEC3 parameters and this is basically an explanation and rationale of where we're going with that. The quick summary is that we're, first of all, not really recommending NSEC3 if you don't need it. If you just have a tiny zone with just a few names, mostly predictable, go with NSEC. It's smaller, faster, better for where it fits well. If you are doing NSEC3, there is too much use of opt-out where it's not needed. I'll talk in more detail about where it might be needed but when in doubt, no, opt-out, and I'll talk about what that is in a moment. I'm also recommending no additional iterations in NSEC3. I think there's some misunderstanding, people might think that zero in the NSEC iterations means no hashing. It means one initial hash and that's quite sufficient in most cases. So when in doubt, use an extra iteration count of zero. And even further, there's no point in salt unless you're regularly rotating it pretty much each time you sign. So at the bottom of this slide, what you're seeing is a hypothetical NSEC3 record which has SHA1 for the algorithm, no opt-out, no additional iterations. Empty salt shown as a minus and then just the usual stuff, the next hash, and some bits.

Okay, so NSEC3, what is NSEC3 anyway? There are two DNS record types associated with NSEC3. One of them is the actual data that provides the denial of existence for a DNS record and it's got an algorithm flags, iterations, salt, and then this next-owner and a type-bitmap. I'm not going to give a tutorial about how this works but these are the fields. The first three fields are shared with the NSEC3 parameter record, also algorithm, flags, iterations, and salt. Our recommendations are basically covering these features, the algorithm, the flags, iterations, and the salts. The NSEC3 parameter is just used primarily between primary and secondary DNS servers and oddly enough in the NSEC3 parameter field the flags are always zero. For some reason that was a choice not to synchronize it with the flags and NSEC3 record.

The algorithm is always SHA1 and my point is that even though we're deprecating SHA1 as a secure cryptographic signature, that's not what it's being used for in NSEC3. It is simply used as a good way to randomize a string for purposes of reducing collisions but it is not a digital signature. It's got the right size for use in NSEC3 and it's plenty secure for NSEC3 and it's fast. So while we're removing SHA1 from almost all other places, we should keep it in NSEC3, it's a very good key fit. There's no reason to switch to anything different. The only flags that are present in the NSEC3 record is the flag one which is opt-out. Again, I'll talk about that in a moment but recommend that you don't use it unless you have to. The iteration count is unfortunately specified as a 16-bit value in the protocol even though really four bits would've been enough and would've kept most people out of trouble iteration count of 0 to about 10 or 16, no problem with that. But 65535 is ridiculously large and should never be used.

In my DANE survey, I found one domain actually with the value set to 65535, definitely not recommended. The salt is there to discourage precomputations that make it faster to then do dictionary attacks against the domain. But the salt is mostly unnecessary because the hashes used in NSEC3 already include each zone's unique domain name so there's no global precomputation that's possible against NSEC3 and so the salt is only useful if you change it every time you're fully resign the zone, and I'll talk about that in a moment. But otherwise, the salt adds no value and can just be kept empty. So, why would you use NSEC3 rather than NSEC? Which I said is a good choice if you don't have a compelling need for NSEC3. So, the first reason to use NSEC3 is that long, long ago back in 2010 when things like the root zone and the .COM zone were being signed .COM was large enough and had few enough signatures that it would be a major burden for them to sign every domain in the .COM zone rather than only the ones that were initially opting in to do DNSSEC.

So memory was expensive, the software wasn't always to sign the zones as quickly as they would like, so they wanted to be able to sign only a fraction of the records in the .COM zone. Opt-out let them sign only the domains that needed to be signed, .COM, by the way, uses no extra iterations and no salt so they're already doing what we're recommending but they do need opt-out and still only by 2.3% of .COM is signed so it still makes a little bit of sense for them to continue with opt-out.

The other barrier to going with NSEC is that people are concerned with zone walking, with the ability of an outsider to essentially numerate every record in your zone by querying them sequentially if you're using NSEC, and okay, so in some cases you don't necessarily want to expose your whole zone to easy enumeration but the point we like to make is by the time you've applied one SHA1 iteration everybody but the most determined attackers will not bother to enumerate your zone anymore.

All of that effort to then go ahead and dictionary attack the SHA1 is much too much work for most people who might want your zone content and they can get most of the data from other sources, it's really not worth it to go the extra mile and try and do dictionary attacks.

The salt can further discourage the precomputation but it's not really particularly effective and we'll write more about that in the Internet draft. If you're interested, you can read. So NSEC3 has its place, as I said for lightly signed zones to keep the burden of signing modest and so that's opt-out and then also to deter casual zone walking. However, you can take NSEC3 too far. So the first thing is that if you don't need opt-out but you turn it on, you're making denial of existence in your zone insecure unnecessarily. So anything that isn't in your zone, somebody can forge records claiming that it exists and DNSEC doesn't protect you against forged creation of insecure delegations. Yeah, that's really all I want to say about that. The other thing is that if your zone is small enough—and by far the majority of the zones that large hosting providers publish have only one or two records and that's it—there's really no reason to attempt to keep anything secret. The zones are small so you don't need opt-out and even in the scale of a few hundred thousand or even a million records in your zone, the memory cost of signing everything is not quite modest so unless you have a very large zone like .COM/.ORG whatever, you don't need opt-out, don't do it.

The other thing about NSEC in terms of taking it too far and the one that we really care about most is that high iterations can really impose CPU burdens on both the authoritative and the recursive servers and because we believe that they serve little purpose in actually protecting your zone against zone walking, we want to strongly discourage aggressively high iteration counts and so we're hoping to persuade the community to bring those numbers down and that's starting to happen now. Keep in mind that if you do want to keep your zone data mostly protected, DNS data leaks in many different ways, it's very unlikely that most of your DNS zone will not be discoverable through other channels. Okay, in terms of salt, again I mentioned earlier that the zone is already hashed, FQDN is already hashed, so there are no global rainbow tables, so a fixed salt—and many TLDs and many other domains have a salt that they never change—it's pointless, it adds a small additional cost to the computation thread in DNSSEC.

If it serves no value, just eliminate it, have an empty salt. If you do change a salt every time you sign, great, go for it, keep it modestly short but it's fine to do that. Keep in mind that if you are introducing a new salt, requires you to regenerate the entire NSEC chain for the whole domain before you can start using it so it essentially has the cost equivalent to a full resigning of the zone. If you're doing incremental signing, you can't do the salt changing in real-time, you do it periodically and it's a somewhat costly operation and that's perhaps one of the reasons why salt changes aren't that frequent. Some people keep the same salt indefinitely. If you really care about zone walking—if that's what you're serious about—instead of relying on NSEC3 and salts and all of that, these days folks like Cloudflare and some others are doing on the fly signing of their DNS zones and when they do denial of existence, their NSEC or NSEC3 records are synthesized to not be actual domains in their zone but to be minimally separated records that just cover the query and nothing else, DNS names that are one bit or one byte, or whatever apart.

And those are quite effective because they give the attacker no information about the zone content at all, and in some cases, like what Cloudflare does with NSEC and no data, the responses are also much smaller, they can get away with returning one signed NSEC record rather than two or three so they keep the response sizes down as well. So if that's what you want to do, then there's technology for it, the only thing is it can only be done if you have the hardware and software and so on to do on the fly signing with something like ECDSA and takes perhaps a little bit more CPU. But once you have the infrastructure for it, it definitely protects your zone against zone walking.

But in most cases, it's unlikely that the zone is worth that effort. Almost all the DNS names that are reused for anything that PRN certificate transparency logs in various passive DNS or for TLDs and CZDS, lots of people get to see a zone content and so on and if you use the dynamic signing with white lies then you lose the ability to take advantage of aggressive negative caching which can reduce load on your authoritative servers. This is not a pitch for doing the various shades of lies but if that's what you really want, then that's certainly an option. In summary, what we're suggesting in the draft is the duration count should really effectively be zero but anything between 0 and 10 is fine. 10 seems to be a popular non-zero value, we're not going to laugh at you if you do 10.

17 TLDs in the last week have switched from numbers above 100 to 10 or less, so [Laos] went from 150 down to 1, .GREEN and .MX from 100 down to 10, and we're hoping to see the 100 or so in one of the clusters that's been observed. There are actually I think 180 domains with 100 iterations, that those will come down as well soon. Opt-out, we want to see less use of that especially in leaf zones. Some operators sign up customers zones that are small and enable opt-out in them, that's a mistake. Don't enable opt-out when you don't need it. Again, this fixed salt, just don't bother, zero length is fine but if you're rotating your salt every time you resign, great. Do it, keep your salt to 16 bytes or less or so and you're in good shape. That's it, and I think we'll do questions after Wes's talk. Right, combined? Unless we want to [inaudible].

FRED BAKER: If you want to defer them until after Wes's talk then we should have Wes talk.

VIKTOR DUKHOVNI: Yeah.

FRED BAKER: Wes, you want to start?

WES HARDAKER: I sure do. I assume you can see my screen, yes?

FRED BAKER: Yes. I can see your screen.

WES HARDAKER: I'm what's known as the closer. I'm what's keeping you between us—and for those on the Pacific coast that are actually still hungry for lunch, for those on the East Coast in the US you probably have eaten—for everybody else in Europe you probably want to go to bed. The Asian continent probably needs more coffee. So, I'm going to talk to you today about being temporarily insecure intentional. It's also referred to as the cheaters' guide to algorithm rolls and the reason that we're diving into this today is there's actually a fair amount of DNSSEC signing algorithms that can be used by keys and there is a list in RFC8624 which is the most recent recommendations for what validators and signing infrastructures should make use of.

So on the right-hand column, you see all of the recommendations for all the algorithms in the first two columns. Note that RSA/MD5 is must not, it's actually been must not for a long time. RSA/SHA1 and SHA1 for NSEC3 are both not recommended now. In other words, if you're using one of these algorithms you should consider switching. RSA/SHA256 is the signing software must support it. Not that you must support it if you're signing your zone, but your software must support it. Then SHA512 is not recommended and part of that is they're just big. GOST is also must not and then there are two elliptic curve ones that are fairly popular which is 13 is the most popular elliptic curve, P256 with SHA256 is a must and we'll see in a minute that that's actually quite popular.

And 15 is also somewhat popular which is ED25519 and that's recommended and then there's these other ones that are mays.

What do they look like today? This is the graphs from the webpage stats.dnssectools.org—that Viktor and I put together—showing the popularity of each of the algorithms. And you can see that RSA/SHA1 which is algorithm five has fallen dramatically and I'll show you more examples of that in a minute, and for good reason. Even algorithm seven which is still fairly popular, which is RSA/SHA1 but with NSEC3 is actually declining and hopefully, it'll continue to decline because we really think that those should go away in favor of the two risers which is RSA/SHA256 and 13 which is the ECDSA curve P256 with SHA256. So, the upshot is you should be in one of those green columns, and if you're not, then you should consider rolling your keys.

Unfortunately, that can seem scary so we'll get to that in a minute. Here's some graphs of the decline of the two older ones, so RSA/SHA1 you see took a cliff dive there, I think that was late last year, it really has fallen out of favor and it's pretty much gone. There's some hold outs and I hope that you're not one of them. I was one of them for a while, just for the record, and then even RSA/SHA1 with NSEC3 is actually slowly declining. That one has a lot further to go so if you're using that one, we strongly suggest you move. Here's a graph of the two risers, so this is 8 and 13 that you saw in the previous graphs of how they have interchanged over the years and they are both fairly popular. I've personally recently switched to 13, they have much smaller signature sizes so they contribute less to DDoS attacks and things like that. Although if you use rate-limiting it's typically okay.

But, so these are the two you that you ought to consider shifting to. So our recommendation is that if you are starting with a new zone today you pick one of those, that I'm sure is pretty obvious to you after my last dialog for a couple of minutes, and note that the signature sizes of RSA/SHA256 are quite a bit larger than the ECDSA one so if you want small packets leaving your network, then you should pick algorithm 13. If you're still using RSA/SHA1 or RSA/SHA1 with NSEC3, it's time to switch to one of those two, but how are you going to do that? So that actually, I think everybody gets very nervous about algorithm roles, for good reason. The right way to do it is you go read RFC6781 section 4.1.4… Actually, the right way to do it is you have software that does this for you, that has read all of this, and knows the right way to do it.

Hopefully, many of you are using reasonable software that actually does algorithm rolling for you automatically. But inside, what they do is they go through this fairly complicated set of steps which is you create new keys with new algorithm but you don't actually publish it yet, you can't publish it yet and there's a bunch of timing considerations that are making you wait for safety to make sure that you never get into a state where some validators with cached data can't validate your zone. So they're really trying to avoid that. You should sign your zones with both of the keys, the old keys, and the new keys, and publish the RRSIGs from both of those but without publishing the key itself. Now, if any of you are using hand-rolled scripts that are using command line tools or something like that, there's no significantly easy way to actually do this.

You have a zone that you need to sign twice with two different keys and then you need to only publish the keys with the older one and then it all gets complex and you end up having to open a text editor if you're using some of these hand-rolled scripts. If you are, hang on we're going to get to your solution in a second. If you're using automated software, they likely take care of this for you, I hope, but you should make sure. And then you finally publish the zone with the new keys after you've waited a couple of TTLs for safety periods and then you end up publishing the new DS records with your parent and removing the old DS record with your parent. Then you wait for the length of the TTS DL, by the way, that's often quite long, watch out for that. It's not the same as your RSSIGs for your DNSKEYs. Then you remove your old DNSKEYs from the zone finally and you wait another TTL before you remove the old RRSIGs.

Again, we're back in this state where the keys that you are publishing actually differ from the RSSIGs that you're including and this all has to do with caching out in the distant wild and that really can be painful again if you're using these hand-scripted things. Here's my solution for you and I just published an IETF document, an Internet draft that's not accepted by the Working Group yet. But I'm hoping that I can actually publish this as an RFC for people to consider doing, but it greatly simplifies all that if you do the absolute wrong thing, you come insecure. And so nobody wants to turn off DNSSEC, but if that's the only way you can get to a new algorithm, my recommendation is you consider doing it anyway. You remove the DS record from your parent, all of a sudden, you're insecure. All of a sudden, no validator can actually find that even though you're publishing keys and data and things like that, you're insecure so that they won't actually look for your keys.

You do have to wait for the DS record to expire. You have to wait for that DS record to flush from all the caches out there so waiting a TTL or two times a TTL is safer, and then you replace your old keys with the new ones right there, right, you're done. You start signing with the new ones, you're done. There's none of that waiting while you have a mixed state of old keys, new keys, old RRSIGs, and new RRSIGs, you just do it, because nobody's actually going to try and validate you while you're in this funky middle state. Then you wait for the zone's negative cache time to expire to make sure that when they're querying for keys that they go away and you've got to wait for, depending on when you remove them you actually wait for the TTL of the RRSIGs as well which is likely bigger.

Then finally you add the new DS record back to the parent and you're secure again, but there's this gap. There's this gap between the point where you go insecure and the point where you come back and you are secure and then you've got to watch out for that. So you know that you are in this point where DNSSEC has been turned off. And you can minimize that time a little bit by lowering the TTLs of your records before performing these steps. Unfortunately, you usually can't lower the DS record in your parent and those are often quite long. Anywhere between four hours and a day even or something like that, so watch out for that. So, why would you do this? Pros and cons. Cheating is operationally much simpler. All of the things I just said, you could script that in a quick batch script in a short period of time and it's less prone to human mistakes. If you're actually doing any of this by hand in order to get to the new algorithm, you're less likely to make mistakes. But it does temporarily transition you out of DNSSEC protection.

So you should really only do this when you're not using any automated software that handles all of this for you and when you're more concerned about stability versus security. So if stability is of paramount importance to you and you're not likely to get DNS attacked in a small window of time, you might consider doing this because it's slightly more stable. So you have to consider what's your threat model, what's your operational model, what should you do? As far as when you should do this, if you're using one of the old algorithms, I think I've made my point clear. Now. Now is the time to do that. And then we're going to get to the question time so I've got you back on time at this point. This is actually a shot from Mexico, I want to say 2008 or something like that, it was actually quite a long time ago but unfortunately, we are not in Cancun today and we're not sitting on a beach and not swinging and having drinks and sharing friends, but we'll get back there at some point. With that, I will turn it back over to Fred for any questions for probably anybody on the panel, to be honest. Fred?

EDWARD LEWIS: Fred's muted?

WES HARDAKER: Fred is muted. I'm reading backwards through some of the questions so…

FRED BAKER: Sorry about that, I stepped away for a moment.

WES HARDAKER: That's okay.

FRED BAKER: But, yeah, you had some questions about DNS speed publication on but only the RRSIG, not the DNSKEY itself.

WES HARDAKER: Yeah, so actually, Ryan was asking when in step two you're waiting for the new key to publish, you should actually roll both keys at once. It's unlikely you're using different algorithms for ZSKs and KSKs, in fact, you really can't. You can use double algorithms but you can't use different KSK and ZSK algorithms safely. There's probably a way to do that and I'm sure Viktor would name it off the top of his head, but you should actually roll both of those algorithms at the same time, so if you have both ZSKs and KSKs, then do them both at the same time. And so all of my slides are really talking both at once and I thought I actually had a note on one of my slides about that, but maybe I never added it. Then, somebody asked what is wrong with parallel signing with two algorithms? Nothing. You can sign with two algorithms but you've got to get the timing right, and Viktor can probably quote the mechanisms for validation. The problem is that some validators out in the world, if they find two DS records, they will mandate that they validate both of them. If either one of them fails, then you're bogus, your zone is bogus.

And the number of those are decreasing and there's some talk in DNS Op about trying to actually remove that nebulous wording of the original DNSSEC specifications, but you can do it, but again you have to do that strange mechanism of publishing RSSIGs first and then signing the two algorithms and then if you want to remove the older one. That's what the point of that is.

VIKTOR DUKHOVNI: Also, if you've got larger RSA keys, there's not a lot of room for a second algorithm there. But if you're going with algorithms 13 and let's say 15, then there's enough room to do that and some people might do that.

FRED BAKER: Okay, there's a question here from Eric Osterweil.

WES HARDAKER: He has his hand up.

ERIC OSTERWEIL: Hey guys. Nice presentation, both of you, that was real interesting, and I'd definitely like to follow up with you guys, read the stuff I was talking about. But the to the point about temporary insecure, Wes, I think just my two cents, it would probably be really useful to talk a lot about what the threat levels are to motivate pro or con. One of the things that occurs to me is that I think, didn't [DNSpionage] do the same thing, strobed DNSSEC off when it wanted to pwn people? So coming up with some way that you might be able to disambiguate whether it's being turned off and that's a bad thing, or it's being turned off and that's an intended thing might be useful. Or I don't know if you have thoughts on that.

WES HARDAKER: I debated that actually for a while because—and I didn't put anything in my Internet draft about it, but setting the TTLs low reduces that window as much as you can. But do you want to advertise it? Do you actually want to give people the window of which you're going to go insecure? Probably not. You probably want to do it, if your attackers are all in North America then you should do it midnight to 4:00 AM North American time or something like that so that there's less people that'll actually get to it. I mean, you're right, you have to consider the whole threat model and whether it's worth doing or not. But my gut instinct personally is that you shouldn't actually advertise that you're going to do it this way because you're opening yourself up to a minor window of attack so you might as well just do it on the sly and hope nobody notices and then come back up and then say you did it.

ERIC OSTERWEIL: Yeah, sorry. I hate to follow my own question with a clarification. Sorry for that. I was just saying some way to disambiguate whether this was on purpose or not, but that's I'm not sure what that means. But just something to think about. Thanks a lot for the response.

WES HARDAKER: Yeah, absolutely.

FRED BAKER: Okay, Ulrich, you have your hand up.

ULRICH WISSER: Yes, hello. Wes, I agree that the DNSSEC RFCs say that you say that you should first publish the signatures and all of that but actually, we have an RFC that says the resolvers should go with lax validation. [inaudible].

ERIC LEWIS: But they don't yet, right?

WES HARDAKER: But they don't yet, right.

ULRICH WISSER: Yeah, they do because…

ERIC LEWIS: Well, most of them do but some don't.

ULRICH WISSER: We actually did roll two TLDs without first publishing signatures, we just double signed immediately and that worked very well and in Sweden, we have over 90% of validation, so…

VIKTOR DUKHOVNI: Right, probably the people for whom it was a problem didn't make enough of a fuss. There are of the small minority. And you're not the only ones who did it. I think recently Israel rolled from 7 to 8 or 13, or whatever and they also just switched from one to the other and they appear to have gotten away with it. But they also think there was some evidence of resolution problems in a few places.

WES HARDAKER: One of the targets of my draft that I tried to indicate on the slides too is that I'm encouraging this for people that want maximum robustness. You don't want any validator on the planet that is continuing to stick to that old model and there's certainly old software out there. So we know that some exists and you're right, that the count is hopefully low and hopefully should be declining. If we get a new RFC publishing everybody shouldn't do that double validation with two DS records then that would certainly pave the way to not needing to do any of this. I think Peter earlier said he's seen people do this as well, and absolutely, rolling through insecure has been done a lot as people are just shifting infrastructure and things like that. It's actually an easy way to temporarily go insecure if you're not doing anything.

So one of the things that I put into the Internet draft are there's a number of places where you probably don't want to do this. If you're making heavy use of TLSA records, you probably don't want to go insecure. If you're making use of DNSSEC validated SSH fingerprints or things like that, you probably don't want to go insecure. Those end up becoming important records that need that security bit turned on.

VIKTOR DUKHOVNI: Also, speaking of Sweden, in fact, there was a large hosting provider in Sweden that back in December of 2019 basically turned off DNSSEC for all of the 100,000 or more of their customer zones over December while they were rolling them to algorithm 13 or so, and so there's a pretty dramatic drop in DNSSEC in Sweden for about a month when that was happening and then they brought it back. So in their case, they decided that going insecure was the right way to do it, there's some precedent for it.

WES HARDAKER: I'll be honest, I did it. I have some manually encoded stuff that was required because I'm pulling data from databases and then doing signing. I found [inaudible] this was an easier way to do it, and so it has been done by me alone.

VIKTOR DUKHOVNI: I'm showing the graph, I don't know if people can see it. I'm sharing my screen. That's the graph for Sweden's DNSSEC and there's December of 2019 or 2018 where it falls off the cliff and then comes back.

WES HARDAKER: Any other questions? There's a Q&A panel that has, "Would you recommend doing this for the root?" No. Good question, David, but no. No, the root has a few customers, just a couple.

FRED BAKER: Just a few. David Conrad has his hand up.

WES HARDAKER: David? Did somebody enable him to speak? Kathy?

DAVID CONRAD: Hey, can you hear me?

KATHY SCHNITT: Yup, just hit it.

DAVID CONRAD: Actually it was a little tongue in cheek since you said you would want to do this if you're primarily interested in robustness and obviously we're primarily interested in robustness at the root. But I understand where you're coming with this. Wasn't actually suggesting we do it for the root.

WES HARDAKER: Well, so you do have to consider the robustness of everything underneath the point.

DAVID CONRAD: Yes.

WES HARDAKER: And so if you have, as I said, people underneath the point that are depending on DNSSEC security—and I can argue, David, that the root has a few people depending on DNSSEC security underneath it—you probably don't want to do it. This should really be at public suffix breakpoints and lower.

DAVID CONRAD: Yeah, understood.

VIKTOR DUKHOVNI: Yeah, doing it at the root would break all kinds of people who have mandatory policies requiring DNSSEC presence for their infrastructure that would break quite quickly.

WES HARDAKER: Yup. All right, I think we're out of questions, Fred.

FRED BAKER: Yeah, we appear to be. So that being the case, Kathy, should I turn this back to you?

KATHY SCHNITT: Yes, thank you Fred. I appreciate that.

VIKTOR DUKHOVNI: I have one comment if anybody's really interested. In an earlier talk, it was mentioned that we're seeing some adoption of ECDSA in TLDs, and my comment was that basically almost all of that is AFNIC, they registered 20 or so domains with ECDSA that they manage ccTLDs. Then, of course, there's Brazil and Czech Republic and a few others. So it's not broad adoption yet. It's just one player made a big move. I'd like to encourage others to start moving TLDs to 13. That would be good. Go ahead, Kathy, sorry.

KATHY SCHNITT: Thank you very much. Well, I want to thank everyone for joining us for the virtual ICANN70 DNSSEC and Security Workshop. I really want to thank all our panelists and moderators for their excellent presentations, the Program and Planning Committee for our fantastic agenda, our amazing tech crew for their assistance, and my amazing colleagues, Kimberly and Andrew for helping me run these sessions so fabulously. Please enjoy the rest of ICANN70, we will see you for ICANN71. We may now stop the recording.

[END OF TRANSCRIPTION]