Ananlysis on Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC And Zone State Revocation for DNSSEC

Padmavathi Dl
pduggire@gmu.edu
George Mason University

ABSTRACT

DNS Security Extensions(DNSSEC) uses public key cryptography to ensure data integrity and origin authenticity in domain name sysytem(DNS). Due to the global scale of DNS, as though the cryptographic design is simple DNSSEC is still in rolling out phase. The proper functioning of internet is critically dependent on DNS, this motivated me to analyse two papers related to DNSSEC. The first paper "Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC"[19] reveals and potrait a fundemental gap between cryptographic designs and operational DNSSEC. To implement a cryptographic protocol in the global scale system such as DNS the protocol should be scalable, flexible, support incremental deployment and function well during imperfect operations. The first paper also provide inputs to design future cryptographic protocols. The second paper "Zone State Revocation for DNSSEC"[10] focuses on a imperfect operation, emergency key revocation mechanism. Zone state revocation(ZSR) is a lightweight and backward compatible enhancement to DNSSEC.

1 INTRODUCTION

One of the Internet's core system is Domain Name System(DNS). For securing this system DNS Security Extensions(DNSSEC) adds cryptographic mechanisms to provide data integrity and origin authenticity of DNS queries. Securing the DNS service not only avoids DNS hijacking and cache poisoning but also provides a idea on how to deploy cryptography based distributed security systems. DNSSEC standards were finalised in March 2005 after efforts by IETF and it is still in rolling out phase because of the DNS global scale. First paper did a good job in not only listing challenges in design, deployment and operational but also explaining the reasons about why the challenges occur. The paper also shows that the problems with DNSSEC is because of large scale and distributed operational DNS system, DNS caching and inherent heterogenity in DNS operations by different autonomous administrators of zones.

After many years of implementation and development efforts, still some DNSSEC deployment issues remain open. Such a issue is emergency key revocation. DNSSEC provides well defined procedures for normal key rollover process where a zone changes its public-private key pairs which occurs periodically but it lacks design when a zone's private key is compromised or lost and it has to be revoked. An adversary can hijack a zone's webservers, namespaces and even create fictious child zones. The main problem is even if the compromised key is removed from the authoritative DNS servers, this does not revoke the compromised key from the caching resolvers. This is because of the global scale DNS there is no way to know which caching resolvers have old keys and old signatures. Also there is no way to notify caching resolvers about the key change because of lack of backward compatability. Adversary can make use of stale data present at caching resolvers until there lifetimes expire. The second paper describes a Zone State Revocation(ZSR) mechanism for handling such emergency key revocation mechanisms. ZSR uses the idea of zone'e state to augment DNSSEC's signing method. With this idea signatures also include state-lease information additional to dates. Because of this change the signatures can be invalidated if a zone's state is changed beyond the specified lease. Also second paper defines a REVKEY record which is also described in first paper to revoke a key and self signed. Second paper describes ZSR offers the below abilities:

- Let a DNS Server mark a public key as being revoked.
- Disseminate information about the revoked key to the large number of resolvers who are distributed at global scale.
- Resolvers can distinguish between key rollovers and emergency key exchanges.

The next following sections of paper is organized as : Section 2 gives background on DNS, DNSSEC and EDNS0. Section 3,4,5 gives a overview of what first paper discussed and also give analysis about the discussion on DESIGN ISSUES WITH A HEIRARCHICAL PKI, IMPACT OF DNS CACHES and IMPACT OF HETEROGENOUS OPERATIONS. In Section 4, in sub section "Key Revocation" I have put my thoughts on second paper. Section 6 gives the analysis on the thought explained by first paper on DNSSEC MONITORING AND DEPLOYMENT STATUS and also the tool SecSpider by authors of first paper. Section 7 concludes the analysis by discussing what are my thoughts on the two papers.

2 BACKGROUND

2.1 DNS and DNSSEC

DNS data is stored at the nameservers in the zone files. The DNS data are stored as Resource Record(RR) in the zone file and each RR have a name, class and type. RRs with same name, class and type are grouped as Resouce Record Set(RRset). If a DNS resolver asks for a RR then the zone sends the whole RRset, which is the smallest unit that the resolver can query for.

DNS Delegation starts with root zone and it delegates authority to top level domains such as .com, .net and .org. Then the top level domains such as .com delegates authority to google.com or .edu to gmu.edu. Each zone is under a single administrative authority and they are identified by “Start of Authority” data record. The Zone’s data can be served by multiple authoritative name servers, primary server holds a master copy of zone’s data and all the secondary servers replicate this data. This synchronization of data is supported by serial number present in the SOA record.
End systems resolve a DNS name by quering the DNS for the corresponding RRset. A end system has a default stub resolver implemented in it, this stub resolver sends queries to caching resolver which does all the complex tasks of forwarding queries to authoritative name servers and sending back the answer after it receives the reply from authoritative name servers.
Since DNS is implemented, a numerous attacks have been identified because security features were not implemented. An adversary can easily exploit these vulnerabiltites to inject spoofed DNS such as Kaminsky attacks. So DNSSEC is used to provide security for DNS data.

2.2 EDNS0

DNS protocol packet format is of fixed size in the early implementation of DNS. But as global scale of DNS increased, it was observed that this fixed size would be exhausted and the client or server cannot inform each other about their capabilities. Because of this problem ENDS option [16]was introduced which will enable UDP format for DNS messages can use the sizes beyond the specified limits for DNS format. During the early implementation of ENDS was an option but over time due to the increased usage of DNSSEC it became a requirement[6]. To allow DNS servers and DNS resolvers to negotiate options such as packet sizes, EDNS0 uses a meta record or OPT psuedo record.

3 DESIGN ISSUES WITH A HEIRARCHICAL PKI

By Using the existing DNS delegation hierarchy DNSSEC implements public key cryptography where each zone signs public key of it’s zones. As though this design sounds simple it ran into problems during deployment. The first paper explained them well as below:

3.1 Scaling

DNSSEC authenticates a zone’s public key by verifying corresponding signature from its parent zone. Previous design of DNSSEC stored this signature at child zone, but this design overlooked the factor that a parent may have many child zones, if a parent changes a key it have to contact all child zones to update new signatures which is a difficult task for top level zones such as .com, .edu. As the child’s zone key has to be changed frequently which needs cross domain interaction between parent and child for the updation of the corresponding signature. Due to above reasons these changes have been made:

- The hash of the child zone key is signed by parent key and stored at parent zone as Delegation Signer Resource Record(RR).
- Zone will have different keys Key Signing Key(KSK) and Zone Signing Key(ZSK).

These changes made DNSSEC scalable and easily deployable.

3.2 Boundary Crossing: Unsigned DNS Data

NS RRs and glue A records of a zone are stored both at parent(so that DNS queries from caching resolvers can perform top-down lookup from root zone to find the intended name server) and also at zone itself. As this data is child’s authoritative data, it is left unsigned at parent zone. However leaving this data unsigned cause no harm when the delegation works perfectly, but if a attacker knows the private key of a child zone: He can forge DNS answers of a zone. He can also forge answers of NS records from parent zone(because this data is unsigned) and redirect queries for child zone to a malicious host. The forged answers will be accepted by resolvers because attacker knows private key of child zone.

Under the above scenario signing NS record of child zone at parent zone adds another layer of protection, but violates data ownership rule because this is the authoritative data of child zone. Here first paper suggests to protect infrastructure records rather than adhering to ownership rule, which I feel is a good suggestion because if a child zone changes its name servers it has to update the same to the parent zone and parent zone can sign it and store in the same way it handles DS RRs.

3.3 Cross - Domain Coordination

As we have seen in the previous boundary crossing NS and glue A records are used in name deleations. So any changes made by child have to be propogated to the parent, these changes are manually updated and so prone to human errors. On average 15 percent of DNS Zones suffer from specific misconfigurations called *lame delegation*, in which parent zone points to wrong name servers for the child zone as shown by the measurement study[12].These configuration errors are not of issue in DNS delegation because of multiple name servers, but a issue in DNSSEC because of periodic key changes. The delegation link works as long as parent knows one correct name server, for this purpose multiple name servers and multiple DS records are present at parent which decreases the cryptographic strength. This is because if attacker breaks any one key matching any of these multiple DS RRs DNSSEC fails.

These configuration errors are explained well in DNS troubleshooting tools[11] and several tools such as [1] are emerged to resolve such errors.

3.4 Incremental Deployability

The individual domains decide whether to deploy new functions so these fucntions can be rolled out incrementally over time. In DNSSEC a zone z data can be verified by authentication chain from the root, if every ancestor of z deployed DNSSEC. If individual domains takes independent decisions to turn on DNSSEC, many islands of security will be formed. Islands of security is a subtree in the DNS hierarchy where DNSSEC have been deployed. The public key of root of such islands of security is called trust anchor. DNSSEC has no implementation for verifying such trust anchor's public key(as parent zone of trust anchor haven't deployed DNSSEC). Below are two proposals for supporting incremental deployment:

- Interconnecting isolated DNSSEC islands through cross-signing where roots of islands signing each other public keys.
- Signing public key of island's root zone by a trusted authorities outside of DNSSEC such as VeriSign called as DNSSEC Lookaside Verification(DLV).

DNSSEC's lack of support for incremental deployment hindered its deployment and several patches have to be yet proven effective, the first paper presents a distributed monitoring solution which we will discuss in section 6.

4 IMPACT OF DNS CACHES

In DNS, caching plays a major role. Stub resolvers implemented at end hosts send direct queries to caching resolvers. The caching resolver first check for the DNS queried data at its own cache, if data is present it will send the cached response to the stub resolver, else it requests data from authoritative name server and return response to stub resolver. Stub resolver doesn't directly communicate with authoritative name server, it only communicates directly with caches. So stub resolver believes what ever data is given by cache resolver is correct. If caching resolver doesn't implement correct security policies an adversary can poison data of cache. When the cache data is poisoned stub resolver might know that the data is invalid but it can't directly communicate with authoritative name server to get the valid data, because of this caching resolver policies and actions have direct impact on stub resolver and authoritative name server. First paper detailed about operations in cache handling as specified in below sub sections:

4.1 Key Rollover

The Zones have to change the keys over time. This means old key have to be phased out and replaced with new key. Due to the caching mechanism even after removing the old key from the zone until the TTLs expired or signature expired it will be present in the caches.
Ignoring the caching effect can break the chain of verification, if a stub resolver receives a old signature and queries caching resolver for key to verify. Let's imagine caching resolver does not have the key then the caching resolver queries authoritative server, then the server replies with new key. Now as the stub resolver have the old signature, it cannot verify with the new key received, it has to reject it until the signature also expires. Below approaches can be used to avoid the above scenario:

- Grace period for the old key during the key rollover process[8].
- An active and a standby KSK. In a key rollover, an active key is retired and standby key will become the active key. And a new standby key is added[14].

Current DNS operation uses second approach as specified in [9].

4.2 Key Revocation

If a private key is compromised it has to be revoked immediately, this scenario is called as "Emergency Key Rollover". One can use the same above key Roll over process for the revocation too, but this change is done in the authoritative name server and so until the signature lifetime expires an attacker can still forge DNS records with the old compromised key. If a private key is compromised the attacker can sign the malicious End Host Records or NameServer Records as discussed in second paper. If the key is compromised these below attacks are possible which are described briefly in second paper.

- Spoofing Attack
- Poisoning Attack
- Man-In-The-Middle Attack - Type 1
- Man-In-The-Middle Attack - Type 2

I feel it is important to know "how an attacker can attack" to stop an attack and authors did a good job in explaining them. For this particular scenario second paper describes a novel approach described below:

4.2.1 Zone State Revocation. ZSR addresses this problem by proposing these changes to the existing DNSSEC implementation.

- Proving Key Compromise - A new type DNS resource record called REVKEY has been introduced which contains the DNSKEY revoked and self signed by it's own private part of the key and storing it in RRSIG.
- Revoking Data - Current DNSSEC validation of resource records is by creating signatures that have TTL of RR, inception and expiration dates (both are combined and a lifetime is created during which the respective signature is considered as valid and validates the Resource Record data). DNSSEC RR lifetimes are dictated by the RRSIGs inception and expiration dates attached to them. As the key emergency situations occur unplanned they will not work well with pre determined lifetimes. ZSR augments the RRSIG inception/expiration values with a numerical lease which will compliments the lifetime of signatures. This lease value indicates whether the zone state has been changed since the signature was generated. If the zone's state or serial number exceeds the value specified in signature the record must be flushed from cache and re fetch the record, also the resolver should query the zone for

any REVKEY records. If an adversary has compromised KSK, secure delegation from the parent zone also have to be broken. The second paper suggests that DS records also have lease concept, so that any DS records stored at caching resolvers can be invalidated. ZSR suggests that the signatures or DS records can have the lease value as current serial number + x. The Zone operators should choose x value such that the normal DNS or DNSSEC operations will not exceed the current serial number + x. For breaking lease zone operator increases it's serial number by 2^{31}. The Second paper have determined this value based on the analysis of sampling of active zones which is discussed in the paper.

- Notifying Resolvers - ZSR disseminates the revocation notifications via lightweight control messages which can be embedded in every DNS response that is only sent from the authoritative nameservers. The control messages are sent in the form of EDNS0 OPT record. There are two mandatory requirements of OPT records : they should not be cached and servers can have arbitrary code/value pairs in them to provide information of their capabilities. So ZSR can provide in the abritrary data this information as a tupule : $\prec$ zone$-$name, serial$-$number, timestamp, signature$\succ$. This signature should be by the key in the REVKEY record, by this way adversary can be stopped from macliciously revoking the key. I found this as a good suggestion because OPT records are already being used as specified in [16] which can be readily used for this scenario also. In this protocol with every query the emergency key revocations can be informed to the caching resolvers. But I have question here what happens if there are no queries from resolvers to the authoritative zones, this scenario is not discussed in the paper. However if there are no notifications then, caching resolver also can see serial number in SOA records, but these records are allowed to be cached and any servers other than authoritative name servers can also responds with SOA records(which can be malicious).

This above revocation mechanism can stop all the attacks listed before. But I feel the above mechanism doesn't work when there are no DNS queries from caching resolver to DNS server. So I have read few other papers to study any other ways I found them complex to understand such as [17] or which may have some issues during implementation. [7] suggests to have Key Revocation List(KRL) RR at the zones and signing them as normal RR but having less TTL, and these KRL has to be signed and stored as hash form and signed at the parent also as List Authentication Resource Record(LA RR). I feel two problems with this method :

- Cache synchronization problem which is discussed in section 4.4.
- This may lead to more configuration errors because of LA RR additon.

4.3 Cache-Stub Verification Policies

First paper detailed why the stub resolver have to trust caching resolver. Different security policies and configurations between stub resolver and caching resolver may result in verification conflicts such as:

- False Negatives - The caching resolver rejects the answers which stub resolver considers valid.
- False Positives - The caching resolver return responses which stub resolver rejects.

DNSSEC allows stub resolver to enforce its own local policy if it doesn't trust caching resolver through a Check Disabled(CD) bit in the DNS query[3]. If the caching resolver receives any query with the CD bit on, it will just forward the answers to stub resolvers and also cache the answers without verification. With this CD bit false negatives can be removed but cannot resolve false positives.The dns answer from the authoritative server is cached by caching resolver. Even if stub resolver considers answer as invalid caching resolver sends the same answer until TTL gets expired. In such situations stub resolver uses different caching resolver or can directly resolve query itself, which defeats DNS caching.

4.4 Cache Synchronization

A cache discards a RRset when TTL expires or the companion signature expires, which ever comes first. When a RRset's signature expires, all the cahing resolvers who hold this expired RRset discards at the same time. If the record is popular such as google.com and facebook.com. then the caching resolvers fetch the RRset at the same time which leads to very high incoming traffic at authoritative name servers which is cache-sync effect. TTL expiration doesn't cause this issue because each caching resolver fetches the RRset data at different times and so the TTl expiration times also will be different. The authoritative zone might sign an entire zone at the same time which exacerbates the cache-sync effect.
First paper developed simple analysis to quantify impact of cache-sync effects. They considered a single A RR served by a authoritative server and fetched by multiple caching resolvers. Queries are sent to the caching resolver by a poisson process with an arrival rate of λ. First paper have said that easily we can see that each cache sends queries at an average rate of (1+λ.TTL), this situation occurs when TTl expires. But I didn't understand how is this given, so I feel that paper haven't clearly explained how they got that average rate. But the peak rate will occur when the signature expires which will be . The imapct of cache-sync effects is shown by the paper as ratio between peak and average load as:

$\gamma= \lambda/\lambda/(1+\lambda.\mathrm{TTL}) = (1+\lambda.\mathrm{TTL})$

Paper recognized such unintended synchronization is not only unique to DNSSEC signature lifetime but also present in reliable muticast design using ACK or NAK which trigger an ACK or NAK implosion at source upon successful delivery or packet loss. Paper made a good conclusion that any protocol designs(or cryptographic designs) must avoid triggering synchronized actions in large-scale distributed systems.

Cache synchronization can be avoided by ttl trimming at caching resolver and replace all signatures before TTL gets expired. As first paper mentioned it is true that instead of these guidelines cache synchronization problem still exists because some zones ignores or forget these guidelines or even after those guidelines due to wrong clock configurations problem still exists.

5 IMPACT OF HETEROGENOUS OPERATIONS

Keeping private keys offline or online of zones is important issue in DNSSEC. If a zone keeps the keys offline for better security protection, two issues arises one is authenticated denial of existence and secure dynamic updates. If a zone keeps the keys online, they impose high security attacks because if an adversary gets hold of this key then can sign the zone data, the impact will be more if an adversary get hold of Key Signing Key intsead(KSK) of Zone Signing Key(ZSK). The first paper wrongly cited [4] where DNSSEC specification recommends the practice of using offline keys, but this is specified in [2].

5.1 Handling Dynamic Updates

Whenever a host moves to a new location it obtains a new IP address by DHCP and use dynamic DNS to change its A record. This update may change the updated RRset itself, the NSEC RRs and the SOA RR and signatures of thise new records. If the key is kept offline, the DNS zone adminsitrator have to invoke offline signer, which will be no more automatic updation mechanism. So the Zone Signing Key must be available to create RRsets which helps dynamic updates [5]. But having this key online is a security risk. But this issue is not because of DNSSEC , but it is the issue because of the operation as suggested by first paper and it kind of make sense.

5.2 Authenticating Denial of Existence

Zone needs to keep the key online to sign and reply answers for nonexistent records. As keeping private key online is not secure NSEC scheme is designed in DNSSEC to construct and sign the proofs of nonexistence. But this mechanism has the problem of zone walking, which cause privacy issues(First paper explained in a detailed way about this). Also NSEC RRs double the size of zone file which leads to high cost. This is a problem for large delegation zones. For the above problem two proposals were made:

- Minimally covered NSEC[18] with keeping private key online.
- NSEC3 without keeping private keys online. But as described in the first paper these over-sized DNSKEY messages which have NSEC3 data can exacerbate PMTU limitations and lead to availability problems for zones. As DNS messages are tranferred over UDP which does not guarantee delivery of datagrams, there is a possibility that one or more fragments of a DNS message will be lost during transfer. With the increase in the number of fragments the number of fragments losts will also increases[13].

As though NSEC3 hardens the problem of zone walking, it does not make it impossible. Using dictionary attacks zone walking is still possible in NSEC3. So there is a recent proposal for NSEC5 which used verifiable random functions (VRFs) to prevent offline enumeration of zone contents and also it does not need keys to be kept online which needs to kept online for NSEC3 white lies[15].

6 DNSSEC MONITORING AND DEPLOYMENT STATUS

For measuring deployment status first paper divided all zones into production zones or operational and test zones. It performed a simple test to determine whether a zone is production or test zone, if any zone is under any Top Level Domains(such as .org, .com) is considered as production zone. At the time of paper was written there were less number of Top Level Domains so it was easy to enumerate them, but these days there are many Top Level Domains and it will be difficult to enumerate I guess in my opinion.
Paper mentions at the time it was written 730 independent islands of security were present, Now the value is around 820000[1]. If all these 820000 islands of security have to configure with trust anchor, managing all 820000 public keys manually will be a difficult task, so the difficulty of this task will be increasing as the islands of security increases. These results are from SecSpider, a tool for dnssec monitoring and deployment status by the authors. This SecSpider is a distributed monitoring tool that issues the same DNSSEC queries simultaneously from multiple locations distributed around the internet. This idea of monitoring is good, but as these multiple locations of pollers is publically available in the SecSpider website, how secure are the pollers and how can one trust that pollers are not compromised, the paper haven't discussed about that. But as paper discussed some level of monitoring is needed in DNSSEC because of the main reason lack of backward compatability. As SecSpider data is available zones, any affected zones can validate their data on the website, the authors specify this as security through publicity which makes sense according to me. Also authors specify this reposiory of data does not work as replcament to DNSSEC PKI, but it helps in incremental deployment to avoid islands of security which makes sense.

7 CONCLUSION

First Paper concludes with a very good conclusion that any cryptographic design face multiple challenges when it is implemented in real world. DNSSEC sound so simple as design but it is so complex when it is implemented and during operation. Not only DNSSEC any global scope design should consider the below designs:

- Design for Scalabiltiy.
- Design for Heterogenity.
- Design for Incremental Deployment
- Design for imperfect Operations

- Design with monitoring as an integral component.

Second Paper provides a simple mechanism for key revocation mechanism which can revoke both ZSK and KSK, but as we discussed in subsection 4.2, I feel this mechanism have some problems.
Before studying and analysing I had a very vague idea of DNS and DNSSEC works. After reading I understood how both works, the problems involved in both, why these problems are and where the problems are. I found first paper very interesting as it covers maximum depth and width of topics DNS and DNSSEC. Also first paper gives many references to understand them more. The second paper gives a proposal of how to resolve key emergency revocation problem involved in DNSSEC which again was so interesting.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to Professor Eric Osterweil for giving a oppurtunity to research on topics such as BGP and DNSSEC which are of global scope and rapidly evolving in security by including this in academics.

REFERENCES

[1] http://secspider.net/.
[2] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. Dns security introduction and requirements,rfc4033. Tech. rep., Mar. 2005.
[3] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. Protocol modifications for the dns security extensions,rfc4305. Tech. rep., Mar. 2005.
[4] Arends, R., Austein, R., Larson, M., Massey, D., and Rose, S. Resource records for the dns security extensions,rfc4034. Tech. rep., Mar. 2005.
[5] Atkins, D., and Austein, R. Dns threat analysis,rfc3833. Tech. rep., Aug. 2004.
[6] Damas, J., Graff, M., and Vixie, P. Extension mechanisms for dns (edns(0)), rfc6891. Tech. rep., Apr. 2013.
[7] Guette, G. Key revocation system for dnssec. Tech. rep., June 2008.
[8] Kolkman, O., and Gieben, R. Dnssec operational practices,rfc4641. Tech. rep., Sept. 2008.
[9] Kolkman, O., Mekking, W., and Gieben, R. Dnssec operational practices, version 2, rfc6781. Tech. rep., Dec. 2012.
[10] Osterweil, E., Pappas, V., Massey, D., and Zhang, L. Zone state revocation for dnssec. Tech. rep., Jan. 2007.
[11] Pappas, V., Fltstrm, P., Massey, D., and Zhang, L. Distributed dns troubleshooting.
[12] Pappas, V., Massey, D., Terzis, A., Zhang, L., Lu, S., and Xu, Z. Impact of configuration errors on dns robustness. Tech. rep., 2004.
[13] Sivaraman, M., Kerr, S., and Song, L. Dns message fragments, internet-draft. Tech. rep., July 2015.
[14] StJohns, M. Automated updates of dns security (dnssec) trust anchors,rfc5011. Tech. rep., Sept. 2007.
[15] Vcelak, J., CZ.NIC, Goldberg, S., Papadopoulos, D., Huque, S., and Lawrence, D. Nsec5, dnssec authenticated denial of existence, draft-vcelak-nsec5-08. Tech. rep., Dec. 2018.
[16] Vixie, P. Extension mechanisms for dns (edns0), rfc2671. Tech. rep., Aug. 1999.
[17] Wang, Z., and Xiao, L. Emergency key rollover in dnssec,ieee. Tech. rep., Sept. 2014.
[18] Weiler, S., and Ihren, J. Minimally covering nsec records and dnssec on-line signing,rfc4470. Tech. rep., Apr. 2006.
[19] Yang, H., Osterweil, E., Massey, D., Lu, S., and Zhang, L. Deploying cryptography in internet-scale systems: A case study on dnssec. Tech. rep., Oct. 2011.