One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in the Internet

Haya Shulman and Michael Waidner,
Fraunhofer Institute for Secure Information Technology SIT

https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/shulman

This paper is included in the Proceedings of the
14th USENIX Symposium on Networked Systems
Design and Implementation (NSDI '17).

March 27–29, 2017 • Boston, MA, USA

ISBN 978-1-931971-37-9

One Key to Sign Them All Considered Vulnerable: Evaluation of DNSSEC in the Internet

Haya Shulman and Michael Waidner
Fraunhofer Institute for Secure Information Technology SIT

Abstract

We perform the first Internet study of the cryptographic security of DNSSEC-signed domains. To that end, we collected $2.1M$ DNSSEC keys for popular signed domains out of these $1.9M$ are RSA keys. We analyse the RSA keys and show that a large fraction of signed domains are using vulnerable keys: 35% are signed with RSA keys that share their moduli with some other domain and 66% use keys that are too short (1024 bit or less) or keys which modulus has a GCD > 1 with the modulus of some other domain. As we show, to a large extent the vulnerabilities are due to poor key generation practices, but also due to potential faulty hardware or software bugs.

The DNSSEC keys collection and analysis is performed on a daily basis with the *DNSSEC Keys Validation Engine* which we developed. The statistics as well as the DNSSEC Keys Validation Engine are made available online, as a service for Internet users.

1 Introduction

Domain Name System (DNS), [RFC1034, RFC1035], is one of the Internet's key protocols, designed to locate resources in the Internet. The correctness and availability of DNS are critical to the security and functionality of the Internet. Initially designed to translate domain names to IP addresses, the DNS infrastructure has evolved into a complex ecosystem, and the complexity of the DNS infrastructure is continuously growing with the expanding range of purposes and client base. DNS is increasingly utilised to facilitate a wide range of applications and constitutes an important building block in the design of scalable network infrastructures.

There is a long history of attacks against DNS, most notably, DNS cache poisoning, [41, 29, 23, 39, 38, 26, 25, 24]. DNS cache poisoning attacks are known to be practiced by governments, e.g., for censorship [1] or for surveillance [28], as well as by cyber criminals. In the course of a DNS cache poisoning attack, the attacker provides spoofed records in DNS responses, in order to redirect the victims to incorrect hosts for credentials theft, malware distribution, censorship and more. To prevent such attacks a number of non-cryptographic defences were standardised [RFC5452].

A number of works developed techniques to bypass the (non cryptographic) DNS defences against cache poisoning. For instance, [30, 25], showed how to predict the source port and transaction identifier (TXID) values used by DNS resolver in DNS requests. The attacker can also apply fragmentation [24] to circumvent the challenge values that would remain in the first fragment. These methods allow off-path attacker to inject valid DNS responses into the communication between a victim DNS resolver and a nameserver. Of course, a man-in-the-middle (MitM) attacker does not need to guess the challenge values: a MitM attacker sees the DNS requests and hence can simply copy the challenge-response authentication parameters from the request to the response. Domains can also be hijacked by exploiting misconfigurations in DNS records, for instance, via manipulation of dangling DNS records (*Dare*), [34].

Recently, [31] performed a study of the DNS caches, evaluating different approaches for injecting DNS records into caches, and overwriting the "already cached" values. In particular, [31] evaluated which records could overwrite the already cached values in victim DNS caches, that would further provide them to the clients. The model of caches was then applied to DNS resolution platforms in the wild, and [31] found that 97% of open resolvers, almost 70% of DNS caches on ISP networks and a bit more than 70% of DNS caches on enterprise networks were vulnerable to at least one type of DNS records injection attacks. The question whether cache poisoing actually occurs in the Internet motivated measurement works which evaluated the fraction of spoofed DNS records in the wild [32, 36, 44].

To mitigate DNS cache poisoning attacks, the IETF designed and standardised Domain Name System Security Extensions (DNSSEC) [RFC4033-RFC4035]. Due to the changes to the DNS infrastructure and to the DNS protocol that DNSSEC requires, its adoption is progressing slowly. Although proposed and standardised already in 1997, DNSSEC is still not widely deployed. Measurements show that currently about 25% of the DNS resolvers validate DNSSEC (e.g., see `stats.labs.apnic.net/dnssec`), which is an increase since few years ago, where about 3% of the DNS resolvers were measured to validate DNSSEC records, [33, 14]. Although only a bit more than 1% of domains are signed with DNSSEC, [47, 27], the situation is improving. More domains are expected to get signed in the near future. This is due to the fact that domains signing is turning into an easy task, as the procedures for automated domains signing by the registrars and hosting providers are becoming widely supported. In particular, ICANN requires that all its accredited registrars and DNS hosting providers support domains signing, encouraging adoption of automated signing procedures, where the DNSSEC keys are generated for customers' domains by the provider and used to sign the DNS records. This process assumes that the registrars and DNS hosting providers are supporting secure and best practices for keys generation.

In this work we perform a study of the vulnerabilities of DNSSEC keys generation in *signed domains*. We detect vulnerabilities and trace the problems to the DNSSEC keys generation practices used by the registrars and DNS hosting providers. The main cause for the problems is twofold: *saving on randomness or faults* during keys generation and *lack of suitable key validation procedures*. The vulnerabilities that we found are not detectable by the online DNSSEC checkers, such as `dnssec-debugger` of Verisign[1] or `DNSViz` of Sandia National Laboratories[2], or DNSSEC command line validation procedures for zonefiles signing.

Our study shows that well established and popular DNS hosting providers and registrars, including those commended by ICANN[3], generate 'weak' keys, for the domains that they host.

Our work is related to a recent study of [22] which showed vulnerabilities in TLS and SSH keys, generated by headless or embedded systems, and server management cards. Subsequently, [22] concluded that the problems are due to faulty implementations that generate keys automatically on first boot without having collected sufficient entropy, and that the problems would not occur on traditional PCs. In contrast, we find vulnerabilities in key generation procedures used by well established hosting providers and registrars (most of which are ICANN accredited[4]), which have the necessary infrastructure to produce the randomness needed for secure generation of DNSSEC keys.

Ultimately our work provides yet another proof that adoption of cryptography in the Internet is a challenging task in practice.

Contributions

In this work we show that even DNSSEC, which is a relatively straightforward application of digital signatures over sets of DNS records, is incurring practical issues in the implementation. This indicates that adoption of cryptography in the Internet is a challenging and error prone task, which requires careful designs and most importantly automation.

We identified vulnerabilities in DNSSEC key generation mechanisms, which exposes signed domains to attacks. To evaluate the scope of the vulnerabilities and the status of DNSSEC adoption we design and implement a framework, we call *DNSSEC keys validation engine*, which allows to collect DNSSEC keys from multiple sources, analyse their security and process them into reports. Our analysis focuses on RSA keys, which as we show in Section 5, constitute a vast majority of the deployed DNSSEC cryptographic algorithms. Our reports quantify the following types of vulnerabilities in signed domains: *keys with shared modulus*, *shared keys* and *weak keys*. The former two categories are comprised of all the signed domains which keys share RSA moduli, either due to use of the same key pair (N,e,d) for all domains or due use of a shared moduli N and different (e,d). The latter category contains signed domains whose keys moduli have a prime factor with 'vulnerable' GCD, i.e., greater than 1 or even, or domains with short (weak) RSA keys. We use our engine to perform Internet-wide collection of $2.1M$ DNSSEC keys[5] used by $900K$ signed domains. We focus on $1.9M$ RSA keys and show that 35% share RSA moduli or are used to sign multiple domains, and 66% fall in the category of 'weak keys', namely, are too short (1024 bit or less) with 0.4% keys being shorter than 768 bit, or have an even GCD. In Section 6.1 we show how to recover the secret signing keys for signed domains which share RSA moduli. In Section 6.2 we apply the fast pairwise GCD algorithm (implemented by [22] and available at `http://factorable.net/`) for all the RSA public keys col-

[1] `dnssec-debugger.verisignlabs.com`

[2] `dnsviz.net`

[3] `https://www.icann.org/resources/pages/deployment-2012-02-25-en`

[4] `http://www.internetsociety.org/deploy360/resources/dnssec-registrars/`

[5] In this work, we report on analysis of keys we collected over a period of March 2016 and September 2016.

lected in order to check whether any of the moduli share a common factor.

We analyse our findings and trace the problems to key generation practices by popular registrars and hosting providers. We discuss good and bad practices for keys generation and derive recommendations based on our study.

We created an online webpage with a list of vulnerable signed domains and DNSSEC keys at `www.dnssec.sit.fraunhofer.de/` (the GitHub project with the code and the data is linked to from the website). Our tool enables the clients to identify secure registrars for signing their domains and alert in case of potential vulnerabilities. We also created an online keys validation service which allows to establish security of the keys during the generation phase, before they are published in public registries and used to sign DNS records.

Organisation

In Section 2 we compare our research to related work. In Section 3 we provide background on DNS and DNSSEC, recap RSA definition and describe two attacks, relevant to our work. In Section 4 we present our DNSSEC-keys validation engine, its components and the data collection that we performed. In Section 5 we perform a measurement of signed domains, then in Section 6 we validate these domains for vulnerable keys, and characterise the factors causing vulnerabilities. We provide countermeasures and describe implementation thereof in Section 7. We conclude this work in Section 8.

2 Related Work

Our work is related to studies on measuring adoption of DNSSEC in the Internet and evaluation of vulnerable cryptography, most notably SSL/TLS. We first compare our evaluation of vulnerable cryptography to the studies conducted in prior work, and then review earlier measurements of adoption of DNSSEC.

Security of cryptographic systems depends on the randomness used in keys generation process. There is a long history of attacks exploiting insufficient randomness in current random number generators, e.g., [5, 7, 9, 12, 20, 21, 16]. There were also attacks applied against systems relying on sources of randomness, e.g., insufficient randomness in pseudorandom number generator was exploited in [30] to predict the TXID values selected by Bind implementations. In 2008, [46] observed that due to an implementation bug the pseudorandom number generator of Debian OpenSSL was predictable.

Adoption of cryptography in the Internet is similarly a challenging task. For instance, the most widely adopted cryptographic mechanism SSL/TLS experienced many attacks due to different implementation faults, e.g., [13, 2]. See [35] for a comprehensive review of vulnerabilities in SSL/TLS.

Most related to our work, is Heninger *et al.* [22] that performed an Internet-wide scan of TLS certificates and SSH keys to analyse security of random number generators (RNGs), and found vulnerabilities in about 1% of TLS certificates and SSH hosts. They showed that the vast majority of the vulnerable hosts are headless and embedded devices, which lack sufficient randomness. In contrast, in our work, we study the registrars and DNS hosting providers, which have the required infrastructure to generate the necessary randomness, yet the fraction of the vulnerable keys in DNSSEC-signed zones that we find is significantly larger than that measured by [22] in keys in TLS and SSH. The results of our work extend the conclusion of [22], in particular, we show that the problem with randomness is not inherent to restricted embedded devices, but is more significant – we show that even large and popular DNS hosting providers and registrars introduce vulnerabilities into multiple signed domains. Our work provides evidence that the problem with the randomness is not only due to resources limited devices but is wider and applies to platforms which possess the necessary means to produce the randomness required for security.

[4] performed factorisation of 1024-bit RSA keys. Some of the keys were vulnerable due to sharing of primes which allowed efficient factorisation via batch GCD computation, while others were factored by taking advantage of randomness generation process.

Previous work on DNSSEC adoption in domains measured the fraction of the *signed* domains, [47, 27], or misconfigurations in signed domains [10, 11], which result in degraded availability. The client side of the DNS infrastructure was also studied, mainly measuring the fraction of *validating resolvers*, [33, 19]. Zone enumeration against NSEC3 was first performed by Bernstein [3] and then Wander *et al.* [45]. Recently Goldberg *et al.* [18] showed that suggestions to improve NSEC3 were also vulnerable to zone enumeration, and proposed [17].

The challenge of performing large scale active measurements of DNS were discussed in [43], which designed and developed an infrastructure for collecting and analysing DNS packets from multiple domains. Due to the large traffic volume that the measurement infrastructure produced (e.g., 123M domains in `com`) and the requirement for repeated data collection on a daily basis, [43] had different latency and storage considerations, than we in our work. In particular, we only focus on $900K$ signed domains. In contrast to [43] we process the results and display them in reports.

3 DNS Security with DNSSEC

In this section we review DNS, and DNSSEC, and then describe the RSA cryptosystem used in DNSSEC and attacks against it (that are relevant to our work).

3.1 Domain Name System (DNS)

Domain Name System (DNS), [RFC1034, RFC1035], is a distributed database containing mappings for resources (also called *resource records (RRs)*), from *domain names* to different values. The most popular and widely used mappings, [15], are for IP addresses, represented by `A` type RRs, that map a domain name to its IPv4 address, and name servers, represented by `NS` type RRs, that map a name server to domain name. The resource records in DNS correspond to the different services run by the organisations and networks, e.g., hosts, servers, network blocks.

The zones are structured hierarchically, with the root zone at the first level, Top Level Domains (TLDs) at the second level, and millions of Second Level Domains (SLDs) at the third level. The IP addresses of the 13 root servers are provided via the *hints* file, or compiled into DNS resolvers software and when a resolver's cache is empty, every resolution process starts at the root. According to the query in the DNS request, the root name server redirects the resolver, via a `referral` response type, to a corresponding TLD, under which the requested resource is located. There are a number of TLDs types, most notably: *country code TLD* (ccTLD), which domains are (typically) assigned to countries, e.g., `us, il, de`, and *generic TLD* (gTLD), whose domains are used by organisations, e.g., `com, org`, and also by US government and military, e.g., `gov, mil`. Domains in SLDs can also be used to further delegate subdomains to other entities, or can be directly managed by the organisations, e.g., as in the case of `ibm.com, google.com`.

3.2 DNS Security Extensions (DNSSEC)

Plain DNS requests and responses are not protected and hence expose the DNS resolvers to DNS cache poisoning attacks, whereby altered DNS records, served by malicious entities, redirect the clients to incorrect hosts. Such attacks can be launched by MitM or off-path attackers. For example, a malicious wireless client can tap the communication of other clients and can respond to their DNS requests with maliciously crafted DNS responses, containing a spoofed IP address, e.g., redirecting the clients to a phishing site.

Domain Name System Security Extensions (DNSSEC) standard [RFC4033-RFC4035] was designed to prevent cache poisoning, by providing *data integrity* and *origin authenticity* via cryptographic digital signatures over DNS resource records. The digital signatures enable the receiving resolver, that supports DNSSEC validation, to verify that the data in a DNS response is the same as the data published in the zone file of the target domain.

• New Resource Records (RRs). DNSSEC defines new RRs in order to store signatures and keys which are then used to authenticate the responses. For example, a type RRSIG record contains a signature authenticating an RR-set, i.e., all mappings of a specific type for a certain domain name. `DNSKEY` is the public-key of a zone, which should be used to verify the signatures on resource records for which the zone is authoritative. The signatures are computed using the corresponding private signing key, and are then stored in `RRSIG` RRs. The private signing key should be kept secret and is recommended to be stored offline (to prevent exposure in case a nameserver is compromised).

To be able to verify that the `DNSKEY` is correct, a resolver also obtains a `DS` RR from the parent zone, which contains a hash of the public key of the child; the resolver accepts the `DNSKEY` of the child as authentic if the value in `DNSKEY` is the same as the (hashed) value in the Delegation Signer `DS` record of the parent. Since the `DS` record of the parent is signed, authenticity is guaranteed.

• Trust Anchor and Chain of Trust. In order to validate keys of (possibly) millions of domains, the resolvers should be preconfigured with the public verification key of the root. For validation of keys of target domains, e.g., `foo.bar`, the resolvers need to establish a chain of trust from the root to the keys of the target domain, by following and validating the keys of the intermediate domains.

A sequence of `DNSKEY` and `DS` RRs form a chain of signed data composed of links between each nodes on the path from the target zone to the root. The `DS` RR authenticates the child's `DNSKEY` at the parent. The authentication starts with a set of verified public keys for the DNS root zone which is the trusted third party. This allows the resolver to construct a chain from the root to the target zone's `DNSKEY`. The public key of the parent is used to validate the signature (in `RRSIG`) on the `DS` RR, which contains the public verification key of the child. This way a link is constructed from the child zone to parent zone. The resolver continues this way until a path from the target zone to the root is established. If there is no valid `DS` RR at the parent zone for the child's `DNSKEY`, then the chain of trust is broken and the resolution is not secure.

• DNSSEC Cryptographic Building Blocks. DNSSEC uses a fixed set of cryptographic algorithms and hash functions[6] with the most used being:

[6]`http://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml`

RSA/MD5, RSA/SHA-1, DSA/SHA-1 [RFC4034], and RSA/SHA-256, RSA/SHA-512 [RFC5702]. As we show in Section 5, the most widely supported algorithm is RSA (with different variations). Hence, in the rest of this section we recap RSA and explain two attacks which we will be using in the following sections to factor the vulnerable keys.

3.3 RSA Security and Attacks

In this section we recap the definition of RSA signatures. We then provide two attacks: a shared modulus attack and factorable modulus attack, that we use in this work. The idea behind digital signatures is to allow the receiver to verify that a transmitted message originated from the sender and was not changed by an attacker. DNSSEC uses RSA to ensure validity and authenticity of DNS records.

3.3.1 Definition

The RSA encryption is the most popular scheme used in DNSSEC. The RSA signatures scheme consists of three procedures, key generation, signing and validation, that are defined as follows:

Key Generation. Upon input 1^n, choose two random prime numbers of length $\frac{n}{2}$ and compute $N = p \cdot q$ (and $\gcd(p,q) = 1$). For $N = p \cdot q$ we have $\varphi(N) = (p-1)(q-1)$. Choose e such that $\gcd(e, \varphi(N)) = 1$, and select d such that $e \cdot d \equiv 1 \mod \varphi(N)$. Output $vk = (e,N)$ and $sk = (d,N)$.

Signing. Given sk and m, compute $\sigma = m^d \mod N$, return (m, σ).

Verification. Given $vk = (e,N)$, a message $m \in \{1,...,N-1\}$ and a signature σ, returns 1, if the signature σ is the correct signature on m (i.e., if $\sigma^e \mod N \equiv m$) and returns 0 otherwise. return $c = m^e \mod N$.

3.3.2 Shared Modulus Attack

Lemma 3.1 (Computing Factors of N) *Given RSA public and private keys (e,N) and (d,N) respectively, there exists an efficient algorithm for computing the prime factors of N.*

We provide a proof for a simplified case of small e (e.g., assume $e = 3$) and refer an interested reader to the general case shown in [6].

Proof 3.2 *Given $e \cdot d = 1 \mod \varphi(N)$ and $\varphi(N) = (p-1)(q-1)$, set $e \cdot d - 1 = c(p-1)(q-1)$ for a constant c. For $1 \le e \le 3$ and $1 \le d \le (p-1)(q-1)$ we obtain that $1 \le c \le 3$. To obtain the exact value of c, we can try each of the three values. As a result, we obtain the following equation: $p + q = N + 1 - \frac{(ed-1)}{c}$. Set $\tau = N + 1 - \frac{(ed-1)}{c}$ and define the following polynomial: $y(x) = (x-p)(x-q) = x^2 - \tau x + N$. Since τ and N are known and p and q are roots of the polynomial, hence findings the roots also gives the factors of N.* □

Lemma 3.3 (Computing secret key d) *Given RSA public and private keys (e,N) and (d,N) respectively, there exists an efficient algorithm that receives e and factors p and q of N and computes d.*

Proof 3.4 *Given p and q compute $\varphi(N) = (p-1)(q-1)$ then calculate $d = e^{-1} \mod \varphi(N)$.* □

We next consider a setting where a number of participants share the same RSA modulus $N = p \cdot q$, but every participant uses a different public and private keys, namely given ψ participants, each participant i ($i \in \{1,...,\psi\}$) uses (e_i, N) as public key and (d_i, N) as the private key. Then, a malicious party A can recover the private signing key of *any* party i and spoof signatures with private signature key d_i.

Given (e_A, N) and d_A party A can compute p, q, and $\varphi(N)$ and then $d_i = e_i^{-1} \mod \varphi(N)$.

Lemma 3.5 (Private Key Recovery) *Assume (e_i, N) is an RSA public key of party i. There exists an efficient algorithm that given e_i and $\varphi(N)$ calculates d_i.*

First, given (e_i, N) and (d_i, N) party P_i can factor N into p and q. Then, given p and q it can compute $\varphi(N)$, and then d_j for any participant j by computing $d_j = e_j^{-1} \mod \varphi(N)$.

Proof 3.6 *Given p, q compute $\varphi(N) = (p-1)(q-1)$ and find d such that $d = e^{-1} \mod \varphi(N)$.* □

3.3.3 Factorable Modulus Attack

In order to explain this attack, we introduce the Greatest Common Divisor (GCD). Let $a, b > 0$ be integers, the GCD of a and b is the largest integer that divides both a and b.

The importance of GCD with respect to RSA is the following: for every prime number p, every number i between 1 and $p-1$, has a GCD of 1 with p, namely, $\forall i \in \{1,...,p-1\}$, $\gcd(p,i)$ (hence has a multiplicative inverse in modulo p).

If two (or more) different RSA keys share a prime factor, then there are three primes p, q, κ in two keys: $N_1 = p \cdot q$, and $N_2 = p \cdot \kappa$. The reuse of a prime factor allows to calculate all the primes: $\gcd(N_1, N_2) = p$, and obtain q and κ as follows, $q = \frac{N_1}{p}$ and $\kappa = \frac{N_2}{p}$. Then one can compute the secret keys.

The GCD can be computed efficiently using Euclid algorithm. Since RSA moduli are comprised of the product of two prime numbers, they have a common factor with another number if and only if they share a prime factor. Therefore, knowing any prime factor of the public moduli, results in complete breakage of the key, since the private key can be directly computed using the factors of the modulus and the public exponent.

4 DNSSEC Keys Validation Engine

In this section we present our framework for collecting and processing DNSSEC keys, illustrated in Figure 1. The challenges of our framework are the following: (1) support of dynamic and easy integration of new data sources, (2) perform efficient and fast data collection from the data sources on a daily basis, (3) updates of keys at expiration, (4) enable online validation of keys, domains and registrars.

In the rest of this section we describe the components of our DNSSEC validation engine, including data sources and data collection with DNSSEC keys crawler (that we call DKrawler). We explain how we address the challenges of our framework and briefly review the analysis of the data and its processing into reports, and their presentation on an online web page.

4.1 Data Sources

Our dataset of signed zones uses the following 'crawling seeds':

(1) the root and Top Level Domain (TLD) zone files – we obtained the root and TLD zone files (e.g., for `com, net, org, info`) from the Internet Corporation for Assigned Names and Numbers (ICANN). From the root zonefile we collected all the TLDs, which at the time of our study contained 1301 distinct TLDs ((`http://www.internic.net/domain/root.zone`). We used the Centralized Zone Data Service (CZDS `czds.icann.org`) to obtain the records in the TLD zonefiles.

(2) we scanned the top-1M popular domains according to Alexa `www.alexa.com`. In this work we refer to Alexa domains as SLDs, although many of the Alexa domains also contain third level domains, they typically belong to the same organisation, i.e., the same SLD.

(3) the `sonar DNS project` [37] – performs a daily scan of IPv4 addresses for a range of Internet protocols as well as a full scan of DNS resource records in all the DNS zones.

4.2 DNSSEC Keys Crawler

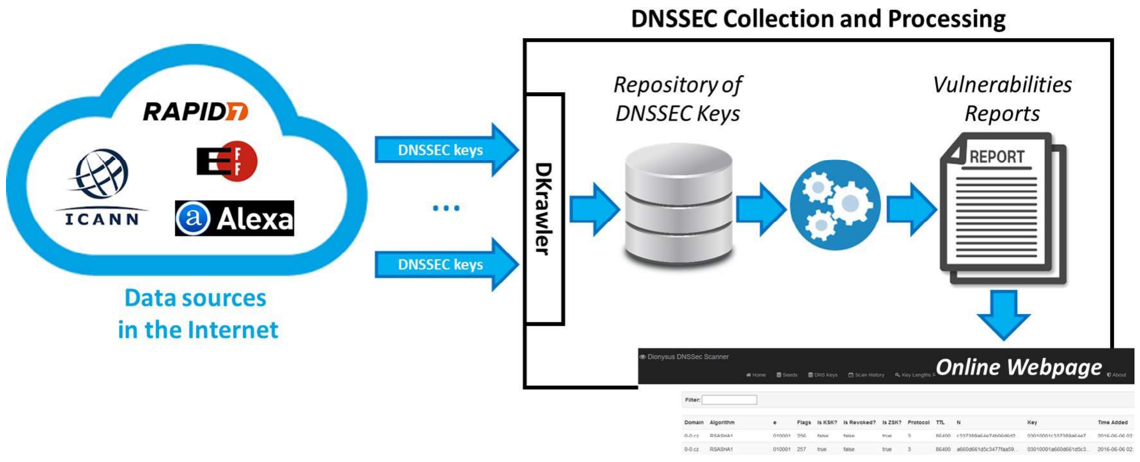
The first challenge is how to find all the DNSSEC-signed domains in order to collect the cryptographic material from them. To that end, we developed a crawler, which we call DKrawler (DNSSEC Keys crawler) to collect and store DNSSEC-signed domains and their keys. The DKrawler infrastructure is written in Python. The DKrawler periodically (every 24 hours) scans the DNS hierarchy using the seeds that it is configured with. New data sources are continually made available to the public, e.g., recently Cisco Umbrella `s3-us-west-1.amazonaws.com/umbrella-static/index.html` made available a repository of popular domains. To support addition of new data sources we designed our architecture in a modular way. This enables easy inclusion of pluggable modules, which we call "scanners". During each scan key material is collected and inserted into the database. Periodical scan ensures that we cover all the new domains that are added.

The DKrawler collects daily $900K$ signed domains using the data sources in Section 4.1. The number of signed domains slightly varies, as new domains are added while some domains become unavailable. The set of $900K$ signed domains result in $2.1M$ DNSKEY records (i.e., DNSSEC keys) out of which $1.9M$ RSA.

Another challenge is efficient traversal of all the data sources. In particular, this includes two main bottlenecks: (1) the waiting time between sending requests and processing the arriving responses, and (2) crawling the DNS servers asynchronously. Sending requests and processing the responses concurrently would facilitate significant reduction of scan time. To that end, we developed the scanners that operate in a non-blocking mode using "greenlets". Greenlets are a light weight version of threads, that can operate concurrently, and in contrast to multi-processing approach, the greenlets are non-preemptive. This allows sending requests and analysing their responses concurrently.

Greenlets also allow to send multiple DNS requests in parallel and to perform the scan much faster while using significantly less resources (in comparison to the multi-processing approach), reducing the memory requirements from the server and the idle time (when waiting for feedback) while speeding up keys collection by more than 30%. We use up to 1024 greenlets, each scans a domain at a given time point. Hence, we scan a total of 1024 domains in concurrently. The responses are processed and analysed, and the statistics are stored in a MongoDB document database, to facilitate fast insertion and retrieval. The processing requires significant memory and storage resources, for instance, intermediate calculations of the pairwise GCD for more than a million keys.

We developed scripts for producing the following reports: *shared modulus keys*, *shared keys*, *weak keys* and *DNSSEC adoption*. We also measure and provide statistics of key length, and popular cryptographic algorithms. We describe the reports along with the collected statistics in Sections 5 and 6. The reports are produced automati-



Domain	Algorithm	e	Flags	Is KSK?	Is Revoked?	Is ZSK?	Protocol	TTL	N	Key	Time Added
0-0.cz	RSASHA1	010001	256	false	false	true	3	86400	c337389a64e74b06d6d2...	03010001c337389a64e7...	2016-06-06 02:
0-0.cz	RSASHA1	010001	257	true	false	true	3	86400	a660d661d5c3477faa59...	03010001a660d661d5c3...	2016-06-06 02:

Figure 1: DNSSEC-Keys security framework.

cally after every daily data collection.

The calculations are efficient, on a two core server, computing the shared moduli report takes up to 3.5 minutes, and the weak keys up to 45 minutes, we describe the implementations in Section 6.

The DKrawler (as well as the entire DNSSEC key validation framework) is deployed on the Microsoft Azure cloud, on a VM machine with 2 virtual (Intel Xeon E5-2673 v3 CPU) cores and 14*GB* RAM memory, each clocked at 2.40GHz.

We have developed an online webpage for reporting the daily surveyed domains and the collected keys (more details below).

4.3 Online Reporting of Vulnerable Keys

We provide access to our tool and the database reports via a website at `www.dnssec.sit.fraunhofer.de`. The code is hosted on GitHub (the link to the GitHub project is on the website `www.dnssec.sit.fraunhofer.de`). The web site is based on a lightweight python web framework ‘Flask’ and MySQL.

The site contains the following tabs: `home` with basic information on scan schedule, seeds and DNS records count; `seeds` lists all the seeds’ sources used to collect the data; `DNS Keys` – lists the collected keys and information, such as protocol, TTL and algorithm; `scan history` – previous scans; `key length report` – lists the different key sizes that we collected; `shared moduli report` and `factorable moduli report`.

5 Evaluating DNSSEC Adoption

In this section we provide our measurement of adoption of DNSSEC among the domains in our dataset (Section 4.1): the Top Level Domains (TLDs) and popular domains according to Alexa (these include second and third level domains). We measure the fraction of signed domains, the key sizes and the cryptographic algorithms. As our measurement results indicate, the majority of domains are signed with different variations of RSA algorithms and almost 50% of the deployed RSA keys are shorter than 1500 bit.

5.1 Quantifying Signed Domains

We define DNSSEC-signed domains as those with `DNSKEY` and `RRSIG` records. To check for the fraction of signed domains, we checked for existence of `DNSKEY` and `RRSIG` records in our dataset. Our finding shows that 87.6% of the TLDs are signed, 1.6% of the Alexa domains; we refer to Alexa domains as SLDs, since third level domains are typically within the same SLDs.

In Figures 2 and 3 we plot the results we collected between March and September 2016. In that time interval the number of new TLDs increased by 100 and we observe roughly the same increase in the number of signed TLDs (see Figure 2). This is consistent with the conclusions of our study in the subsequent section, where we show that most registrars and DNS hosting providers perform automated signing of newly registered domains.

We observe a growth in a number of new Second-Level Domains (SLD) domains, however, in contrast to the steady increase in signed TLDs the results indicate a negligible increase in newly signed domains, see plot in Figure 3. The significant and constant growth in the number of signed TLDs indicates that there is an increased awareness to DNSSEC adoption. This increase is also perhaps due to the automated signing adopted by the registrars. In the rest of this work, we study the

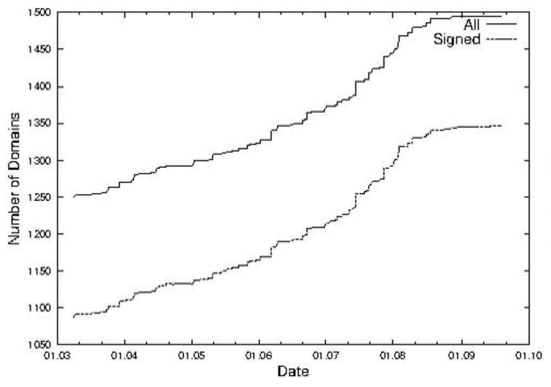


Figure 2: All TLDs and signed TLDs.

signed domains among TLDs and Alexa domains (i.e., SLDs).

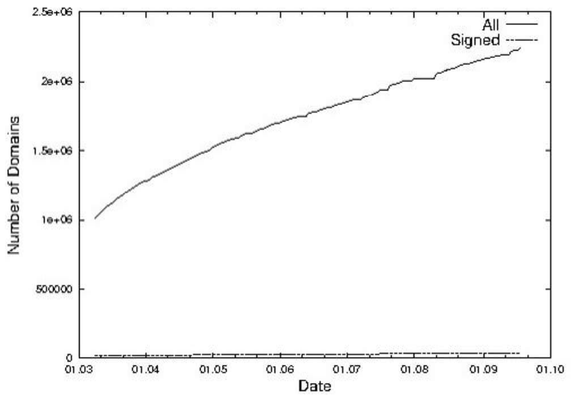


Figure 3: All SLDs and signed SLDs.

5.2 Crypto-Algorithms in Signed Domains

The signed zones can use an arbitrary number of DNSSEC-standardised algorithms (see overview of DNSSEC in Section 3.2). In addition, [RFC4641,RFC6781] list mandatory support for RSA and recommend avoiding large keys (specifying a range of 512-2048 bits for (ZSK) key size and recommending a default value of 1024 bits); in order to avoid fragmentation, communication and computation overhead and other problems with large keys and signatures. In particular, [RFC6781] states “it is estimated that most zones can safely use 1024-bit keys for at least the next ten years.”.

Our measurement of DNSSEC adoption among signed domains shows that there is hardly any support for other cryptographic algorithms, e.g., those that produce short signatures, such as ECC, since the motivation to add more overhead to the transmitted data is low. Our results show that RSA, with different digest implementations (SHA1, SHA256, SHA512), dominates among the signed TLDs, and that there is no support for other algorithms among the TLDs, Figure 4. In contrast, there

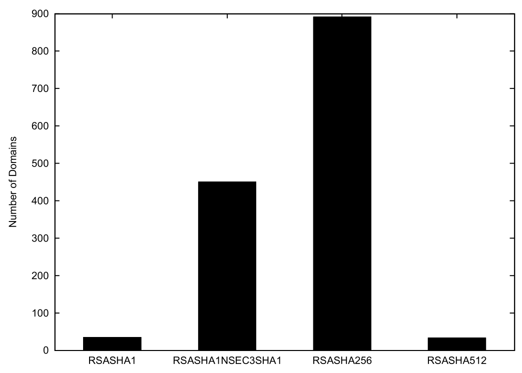


Figure 4: DNSSEC algorithms in signed TLDs.

is some, albeit still limited, attempt to adopt also other cryptographic algorithms, such as DSA and EC in SLDs. These constitute a bit more than 10% of the deployed cryptographic algorithms in DNSSEC, see Figure 5.

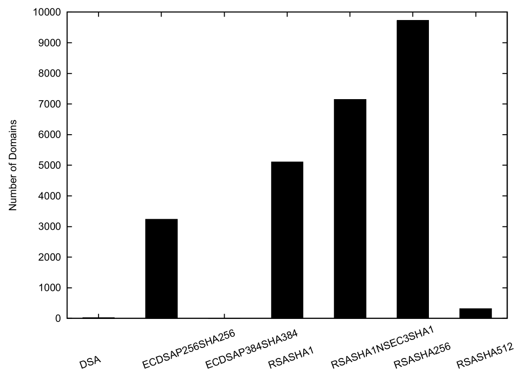


Figure 5: DNSSEC algorithms in signed SLDs.

We measured the key sizes in use by the different variations of RSA algorithms, we plot our results in Figure 6. Almost 1.4M keys are below or equal to 1024 bits, and 10K keys are 512 bits long; [42] showed that factoring 512 bit keys on a cloud is a practical task. For updated statistics on keys and DNSSEC algorithms see our webpage `www.dnssec.sit.fraunhofer.de/key_lengths`.

In a recent work [8] also showed that there are 1% of domains among TLDs and 20% among the SLDs to which it is not possible to establish a chain of trust from the root. The problems include wrong (or missing) DS

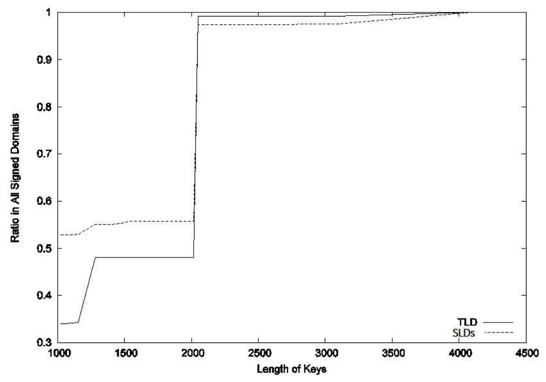


Figure 6: Key sizes in TLDs and SLDs.

records in parent domain, incorrect (or missing) signatures, expired keys, DNSKEY and DS do not match and more. The most common case of broken chain of trust is an existence of DNSKEY but no DS in parent. This may happen when a domain owner wants to enable DNSSEC but his registrar does not support DNSSEC, which is common. Alternately, the same obstacle occurs when the registrar does not support DNSSEC for a TLD under which the domain is registered, e.g., GoDaddy supports DNSSEC only for 10 TLDs. Other common cause is a faulty DS record. This may happen when the domain operator transfers/updates the domain/key or changes the name servers. Broken chain of trust can sometimes be an indication of signed domains in test phase, experiments and deployment in progress.

6 Vulnerable Signed Domains

In this section we describe our measurements of DNSSEC keys generation practices supported by registrars and DNS hosting providers and analysis of vulnerabilities based on the data we collected. We discuss how keys with shared moduli enable recovery of secret signing keys of one domain using the keys of attacker's domain, we show attacks on keys with vulnerable GCD and domains signed with the same keys.

6.1 Shared Keys and Shared Moduli

We differentiate between two cases of vulnerabilities in signed domains: keys with shared moduli and identical keys. In the former case the public validation key e is different, while domains in the latter category have the same N and (e,d). Both practices are vulnerable, in particular, *recovering a key pair for one domain, exposes all other domains signed with the same key or the same modulus, to attacks.*

The attack against domains that are signed using the exact same key pair is straightforward. In particular, once the attacker compromises the key pair, all the domains signed with that key are vulnerable to hijacking. The attacker can generate signatures that will be accepted as valid. Since the records will be signed, the resolvers will assign high trust level to them, allowing to overwrite the previously cached values, [31].

Alternately, it may appear that 'just' reusing the modulus between signed domains does not introduce a vulnerability. We explain that given a key (N,e_1,d_1) the attacker can recover d_2 that belongs to key (N,e_2,d_2). In order to calculate the secret signing keys the attacker needs the knowledge of $\varphi(N)$ (see Section 3.3.2) which it does not have and cannot obtain by querying the DNS information from the domains. To that end, the attacker needs to register and sign a domain under the registrar (or DNS hosting provider), which is used by the (victim) domains whose private key the attacker wishes to recover. Registering domains under most registrars typically costs up to 10$ per year. After the registrar or the hosting provider signs the domain of the attacker, given the key pair which was used to sign its own domain, the attacker computes $\varphi(N)$ and uses the calculation in Section 3.3.2 in order to recover the private signing keys of other domains signed by that registrar (resp. hosting provider). Not all the registrars provide the key pair to the domain owner. In Section 6.1.2 we discuss the countermeasure of keeping the key secret from the domain owner.

Through collection and analysis of keys we found that 700K of the keys were used to sign domains in ways exposing the domains to attacks above. We used whois to retrieve registrars' information for domains signed with shared-moduli-keys. We collected the values of the following fields (consider for example reg-centrum registrar):

```
registrar:   REG-CENTRUM
address:     CZ
IP location: Prague
ASN:         AS43614
```

We grouped the domains according to registrars and according to shared-moduli-keys. The resulting groups were similar. Namely, the shared-moduli-keys were typically found within domains under the same registrar. On the other hand, domains signed with shared-moduli-keys are owned by different operators. This suggests that the registrars, generating the DNSSEC keys, are reusing the keys among the domains that they sign.

We registered and signed domains under selected registrars, which had a large fraction of shared-moduli-keys among signed domains. The registrars typically generate the DNSSEC keys on their platforms, then use them to sign the DNS records. The corresponding Delegation

Signer (DS) records are derived from the DNSSEC keys and are updated in the parent domain (typically a TLD).

For example, in one large and popular registrar over $63K$ different domains are signed with two keys ($51,000$ domains are associated with one of the keys and $12,000$ with the other).

We describe our approach for collecting signed domains with shared-moduli-keys.

6.1.1 Finding Domains with Shared Moduli

Given a list of domains, our tool first collects all the DNSKEY records for signed domains. Then, the tool iterates over all the domains and creates a mapping between each modulus and domains that use that modulus. The tool checks if the public verification keys of the signed domains are different or the same. The domains are signed with a single key when the public e are identical.

```
collect-keys(List):
  For each domain D in List:
key = dns.request(DNSKEY)
    store-in-DB(key)

map-modulus(List):
  d = new dictionary()
  For each domain D in List:
   modulus = D.modulus
 d[modulus].append(D)
```

At the completion, `d` contains a mapping between the different moduli and domains that use that moduli. To extract a list of domains that share a modulus we iterate over `d` and collect domains with length greater than 1. In Figure 7 we plot the results of our calculations of domains with shared-moduli-key over a period of March-September 2016. The graph shows that the fraction of signed domains with shared-moduli-keys is stable over time, and slightly increases as new domains are signed. The majority of moduli and keys are shared among few domains (approximately 10,000). Alternately, few modulus values are shared among a large fraction of signed domains (almost 14%). These moduli are associated with signed domains operated by large registrars.

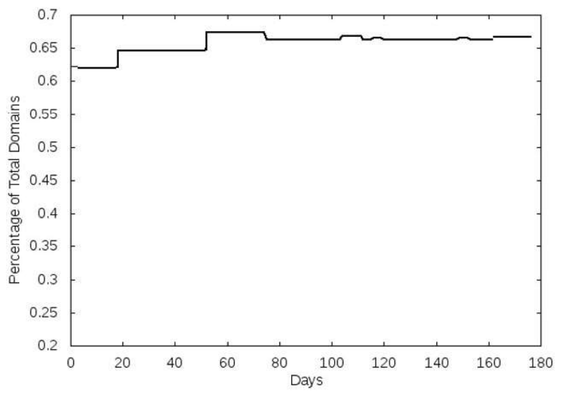


Figure 7: March-September 2016 measurements of fraction of domains with shared moduli (count per day).

6.1.2 Hiding Secret Keys as a Countermeasure

If the registrars, supporting DNSSEC, allow domain owners to obtain the keying material pertaining to their signed domains, the attackers can use this to attack domains signed with the same keys or signed with keys that share modulus values. In what follows we consider whether withholding the crpytographic material would be a viable solution against the vulnerabilities.

The RSA modulus is advertised as part of the public key, and is needed for verification of DNSSEC signatures. According to the Lemmas in Section 3.3.2, given a secret key of one domain, an attacker can recover a secret signing key of another domain (that is signed under the same RSA modulus). Hence it may appear that to counter the vulnerabilities due to a shared RSA modulus, a registrar should hide the secret signing keys from the owners of signed domains.

As we next argue, a naive countermeasure of hiding the RSA secret keys from the domain owner does not constitute a good defence and, moreover, is often not practical. First and foremost, using a single key for singing multiple domains, while keeping it secret from the domain owners, leads to an insecure practice. In particular, *a compromise of the secret key, e.g., via compromise of a registrar, would lead to compromise of the security of all the domains which were signed using that key*. Compromises of registrars are common, e.g., [48].

Hiding the keys could also limit the domain owners to using only the nameservers of one specific registrar - the one that performed the DNSSEC signing. This stands in contrast to the best practices of DNS - which recommend maintaining servers under at least two domains. For instance, consider a scenario where a domain owner registers a domain, which gets signed by the registrar, and decides to use the nameservers provided by that registrar. When the operator adds its own nameserver, and configures as master, he would need a secret signing key to sign the zone file also on its own nameserver.

Our study of ICANN accredited registrars shows that none of the registrars that support DNSSEC, enable it for all TLDs. For instance, GoDaddy enables DNSSEC only for ten TLDs. As a result, if the domain owner wishes to follow best practices and place its domain under two TLDs, it will often not be able to, if it does not have the secret signing key used by one of the registrars.

6.2 Even Moduli & Keys w/Shared Primes

Distinct moduli that share a prime factor will result in public keys that appear distinct but whose private keys are efficiently computable by calculating the greatest common divisor (GCD). For calculation of GCD of every pair of keys we followed the approach in [22] and used the *fast pairwise GCD* quasilinear-time algorithm for factoring a collection of integers into coprimes; we compiled and used the source code (`https://factorable.net/resources.html`) provided by [22].

The code implements an efficient (quasilinear time) algorithm to compute the GCD of every pair of integers.

After calculating group-GCD on all the DNSKEY records, we found that out of factorable moduli 16 RSA moduli were even. Re-querying the nameservers multiple times for these moduli returned the same results over the period of our measurement. Among the possible causes for this could be either a faulty key generation or bit-errors, e.g., due to heat, on the machines generating and storing them. This situation also indicates no DNS servers (nor the platforms operated by the registrars and DNS hosting providers) validate the correctness of the generated DNSKEY records.

The keys with even RSA moduli belonged to domains hosted or registered by known registrars, such as Network Solutions, GoDaddy, OnlineNic. In Figure 8 we plot our measurements of factorable RSA keys, collected over a period of March-September 2016.

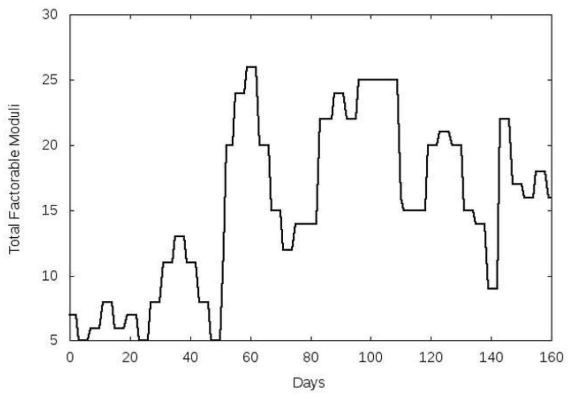


Figure 8: March-September 2016 measurements of fraction of domains with factorable moduli (count per day) as a function of the total number of keys.

6.3 Poisoning Signed DNS Records

We next describe steps for poisoning caches of DNSSEC-validating victim resolvers with spoofed DNS records for signed domains. Assume the attacker at IP `6.6.6.6` wishes to hijack emails sent by clients on network `1.2.3.0/24` to an email server in domain `vic.tim`, and assume that the domain `vic.tim` is signed with DNSSEC. Further, assume that `vic.tim` is hosted with registrar Reg that uses the same modulus N for all its keys. In step (1) the attacker registers and signs a domain `att.ack` with Reg. (2) the attacker uses the secret signing key and the modulus (as in Section 6.1) to calculate the secret signing key for `vic.tim`. (3) the attacker creates a valid signature over a spoofed record pointing an email server at `vic.tim` to IP 6.6.6.6. (4) the attacker performs cache poisoning attack injecting a signed record `mail.vic.tim A 6.6.6.6`. Notice that DNSSEC signed DNS records are assigned the highest trust level by the caches, and hence overwrite any other previously cached value associated with the record.

6.4 Discussion

What are the causes for the phenomenon of shared keys and moduli in DNSSEC keys? The answer is perhaps related to the recent ICANN regulation, [40], requiring that the registrars and DNS hosting providers support DNSSEC for all domains they host or register. On the one hand, the registrars and DNS hosting providers are required to offer and support DNSSEC, and to automate the signing and keys generation processes for the customers. A notable example for this is Binero, which performs the keys generation and signing by itself. Automation of DNSSEC certainly facilitates faster deployment thereof among unsigned domains. The regulation is important as it quickly increased the fraction of domains adopting DNSSEC. On the other hand, automating keys generation requires integrating the necessary software, support on the web interface, access to sources of randomness and using suitable hardware and processes. The registrars and DNS hosting providers may be tempted to save on randomness, and to reuse keys among multiple domains that they sign. As we showed, this practice is vulnerable and can potentially lead to an *illusion of security*, while rendering the *signed* domains exposed to key recovery attacks, and their clients (i.e., DNSSEC validating resolvers) to DNS cache poisoning.

Automating adoption of cryptography is an important goal, and a notable effort undertaken by ICANN [40], however, to facilitate it, the registrars and DNS hosting providers should be equipped with suitable procedures to provide secure services to their customers.

7 Countermeasures and Defences

The community spent a considerable effort to design and operate tools for validation of DNSSEC. For instance Verisign provides `http://dnssec-debugger.verisignlabs.com/`, Sandia

National Liboratories operate `http://dnsviz.net/`, the French operator AFNIC operate the Zonemaster `http://zonemaster.net/`. Upon input a domain name, these tools perform analyses of the configuration of the domain and deployment of DNSSEC, e.g., correctness of the signatures or ability to establish a path to the trust anchor. Since these tools operate on each domain in isolation they cannot detect keys or moduli sharing.

Another set of tools validate correctness of zonefile signing. In particular, they verify that the keys of a target domain sign the DNS records, and that all the required DNSSEC records (such as NSEC or NSEC3) are present in the zonefile. These tools enable domain owners to identify a broken chain of trust.

Unfortunately, these tools do not detect *vulnerable* keys, such as those with a shared or factorable moduli. Tools for detection of vulnerabilities in cryptographic keys would enable the registrars and DNS hosting providers to identify the problems in the keys that they generate, hence preventing the faulty keys from being stored and served by the DNS servers.

Our recommendation is to prohibit sharing of DNSSEC keys across multiple domains. Such a regulation could be overseen by The Internet Corporation for Assigned Names and Numbers ICANN (`www.icann.org`) or Internet Assigned Numbers Authority IANA (`iana.org/`). However, even if put forth, such policies cannot be validated without tools. In what follows we describe two aspects of our DNSSEC-Keys validation engine: one allows online validation of generated keys, the other produces lists of vulnerable domains signed with shared keys. Both validations are integrated into our framework (Section 4). We explain the defences and refer an interested reader to the online webpage of our framework (`www.dnssec.sit.fraunhofer.de/`).

Online Keys Validation Service

Our DNSSEC-keys validation engine (Section 4) supports validation of keys and of signed domains. Given a key in an input, we automatically perform the following checks: (1) the RSA modulus is not even; (2) the new modulus does not collide with the stored modulus values of the scanned domains; (3) the new modulus does not share prime factors with the stored moduli values of the scanned domains. Performing such a validation requires comparing each newly generated key to the keys of (multiple other) signed domains. By comparing a given key to all other keys we verify that the key does not share a modulus with other domains and that it is not composed of prime factors shared with keys in other domains.

This service can be used by domain owners as well as by registrars to ensure security of generated keys.

Listing Vulnerable Domains

We keep a dynamic list of vulnerable domains signed with same keys or domains signed with keys sharing modulus values. This enables customers to evaluate the security offered by the different registrars and hosting services.

Such lists can also be used by firewalls to trigger alerts when clients access vulnerable domains, or by browsers (via a simple browser extension) to alert web clients about a potential security risk, since the domain is not signed.

8 Conclusions

While SSL and TLS received a significant attention from the research and operational communities, and prior work measured vulnerabilities in crypto-algorithms in the wild, the adoption of cryptography used by DNSSEC-signed domains requires more attention. In this work we perform an Internet-wide study of keys in signed DNSSEC domains. To that end, we first measure adoption of DNSSEC and collect a dataset of RSA signed domains. We then validate the security of the keys in our collected set of signed domains. Our results indicate that multiple domains are signed with shared keys. The vulnerabilities stem from poor key generation practices used by registrars and DNS hosting operators.

We developed a DNSSEC keys validation engine, to periodically collect, analyse and process the DNSSEC keys used by TLDs and other popular domains. We set up a website for online reporting of the analyses over the collected data, as a service to the community. Our online service supports validation of keys, enabling clients to identify vulnerabilities in newly generated keys, before the keys are used to sign the zone files and published in online repositories.

9 Acknowledgements

We thank Gal Beniamini and Matan Ben Yosef for developing the tool for collection and processing of keys and Stephane Bortzmeyer, Amit Klein and Roy Arends for their helpful comments on our manuscript. We are also grateful to the anonymous referees for their thoughtful feedback on our work. The research reported in this paper has been supported in part by the German Federal Ministry of Education and Research (BMBF) and by the Hessian Ministry of Science and the Arts within CRISP (www.crisp-da.de/). We are grateful to Microsoft Azure Research Award, which provided hosting for our research infrastructure.

References

[1] D. Anderson. Splinternet behind the great firewall of china. *Queue*, 10(11):40, 2012.

[2] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, et al. Drown: Breaking tls using sslv2.

[3] D. J. Bernstein. Nsec3 Walker. http://dnscurve.org/nsec3walker.html, 2011.

[4] D. J. Bernstein, Y.-A. Chang, C.-M. Cheng, L.-P. Chou, N. Heninger, T. Lange, and N. Van Someren. Factoring rsa keys from certified smart cards: Coppersmith in the wild. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 341–360. Springer, 2013.

[5] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM journal on Computing*, 13(4):850–864, 1984.

[6] D. Boneh et al. Twenty years of attacks on the rsa cryptosystem. *Notices of the AMS*, 46(2):203–213, 1999.

[7] B. Chor and O. Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[8] T. Dai, H. Shulman, and M. Waidner. Dnssec misconfigurations in popular domains. In *International Conference on Cryptology and Network Security*, pages 651–660. Springer, 2016.

[9] D. Davis, R. Ihaka, and P. Fenstermacher. Cryptographic randomness from air turbulence in disk drives. In *Advances in CryptologyCRYPTO94*, pages 114–120. Springer, 1994.

[10] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra. A case for comprehensive dnssec monitoring and analysis tools. *Proceedings of SATIN*, 2011.

[11] C. Deccio, J. Sedayao, K. Kant, and P. Mohapatra. Quantifying and improving dnssec availability. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–7. IEEE, 2011.

[12] L. Dorrendorf, Z. Gutterman, and B. Pinkas. Cryptanalysis of the windows random number generator. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 476–485. ACM, 2007.

[13] Z. Durumeric, J. Kasten, D. Adrian, J. A. Halderman, M. Bailey, F. Li, N. Weaver, J. Amann, J. Beekman, M. Payer, et al. The Matter of Heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 475–488. ACM, 2014.

[14] K. Fukuda, S. Sato, and T. Mitamura. A technique for counting dnssec validators. In *INFOCOM, 2013 Proceedings IEEE*, pages 80–84. IEEE, 2013.

[15] H. Gao, V. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, and H. Duan. An empirical reexamination of global dns behavior. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 267–278. ACM, 2013.

[16] I. Goldberg and D. Wagner. Randomness and the netscape browser. *Dr Dobb's Journal-Software Tools for the Professional Programmer*, 21(1):66–71, 1996.

[17] S. Goldberg, M. Naor, D. Papadopoulos, L. Reyzin, S. Vasant, and A. Ziv. Nsec5: Provably preventing dnssec zone enumeration. *IACR Cryptology ePrint Archive*, 2014:582, 2014.

[18] S. Goldberg, M. Naor, D. Papadopoulos, L. Reyzin, S. Vasant, and A. Ziv. Stretching nsec3 to the limit: Efficient zone enumeration attacks on nsec3 variants. 2015.

[19] O. Gudmundsson and S. D. Crocker. Observing DNSSEC Validation in the Wild. In *SATIN*, March 2011.

[20] P. Gutmann. Software generation of random numbers for cryptographic purposes. In *Proceedings of the 1998 Usenix Security Symposium*, pages 243–257, 1998.

[21] Z. Gutterman, B. Pinkas, and T. Reinman. Analysis of the linux random number generator. In *Security and Privacy, 2006 IEEE Symposium on*, pages 15–pp. IEEE, 2006.

[22] N. Heninger, Z. Durumeric, E. Wustrow, and J. A. Halderman. Mining your ps and qs: Detection of widespread weak keys in network devices. In *Presented as part of the 21st USENIX Security Symposium (USENIX Security 12)*, pages 205–220, 2012.

[23] A. Herzberg and H. Shulman. Security of patched DNS. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, pages 271–288, 2012.

[24] A. Herzberg and H. Shulman. Fragmentation Considered Poisonous: or one-domain-to-rule-them-all.org. In *IEEE CNS 2013. The Conference on Communications and Network Security, Washington, D.C., U.S.* IEEE, 2013.

[25] A. Herzberg and H. Shulman. Socket Overloading for Fun and Cache Poisoning. In C. N. P. Jr., editor, *ACM Annual Computer Security Applications Conference (ACM ACSAC), New Orleans, Louisiana, U.S.*, December 2013.

[26] A. Herzberg and H. Shulman. Vulnerable delegation of DNS resolution. In *Computer Security - ESORICS 2013 - 18th European Symposium on Research in Computer Security, Egham, UK, September 9-13, 2013. Proceedings*, pages 219–236, 2013.

[27] A. Herzberg and H. Shulman. Retrofitting Security into Network Protocols: The Case of DNSSEC. *Internet Computing, IEEE*, 18(1):66–71, 2014.

[28] M. Hu. Taxonomy of the snowden disclosures. *Wash & Lee L. Rev.*, 72:1679–1989, 2015.

[29] D. Kaminsky. It's the End of the Cache As We Know It. In *Black Hat conference*, August 2008. `http://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-Kaminsky/BlackHat-Japan-08-Kaminsky-DNS08-BlackOps.pdf`.

[30] A. Klein. BIND 9 DNS cache poisoning. Report, Trusteer, Ltd., 3 Hayetzira Street, Ramat Gan 52521, Israel, 2007.

[31] A. Klein, H. Shulman, and M. Waidner. Internet-Wide Study of DNS Cache Injections. In *INFOCOM*, 2017.

[32] P. Levis. The collateral damage of internet censorship by dns injection. *ACM SIGCOMM Computer Communication Review*, 42(3), 2012.

[33] W. Lian, E. Rescorla, H. Shacham, and S. Savage. Measuring the Practical Impact of DNSSEC Deployment. In *Proceedings of USENIX Security*, 2013.

[34] D. Liu, S. Hao, and H. Wang. All your DNS records point to us: Understanding the security threats of dangling DNS records. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1414–1425, 2016.

[35] C. Meyer and J. Schwenk. SoK: Lessons learned from SSL/TLS attacks. In *International Workshop on Information Security Applications*, pages 189–209. Springer, 2013.

[36] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman. Assessing dns vulnerability to record injection. In *Passive and Active Measurement*, pages 214–223. Springer, 2014.

[37] security research project by Rapid7. Project Sonar, 2013-2016.

[38] H. Shulman and M. Waidner. Fragmentation Considered Leaking: Port Inference for DNS Poisoning. In *Applied Cryptography and Network Security (ACNS), Lausanne, Switzerland*. Springer, 2014.

[39] H. Shulman and M. Waidner. Towards security of internet naming infrastructure. In *European Symposium on Research in Computer Security*, pages 3–22. Springer, 2015.

[40] I. Society. ICANNs 2013 RAA Requires Domain Name Registrars To Support DNSSEC, 2013.

[41] J. Stewart. Dns cache poisoning–the next generation, 2003.

[42] L. Valenta, S. Cohney, A. Liao, J. Fried, S. Bodduluri, and N. Heninger. Factoring as a service.

[43] R. van Rijswijk-Deij, M. Jonker, A. Sperotto, and A. Pras. A high-performance, scalable infrastructure for large-scale active dns measurements. *IEEE Journal on Selected Areas in Communications*, 34(6):1877–1888, 2016.

[44] M. Wander, C. Boelmann, L. Schwittmann, and T. Weis. Measurement of globally visible dns injection. *Access, IEEE*, 2:526–536, 2014.

[45] M. Wander, L. Schwittmann, C. Boelmann, and T. Weis. Gpu-based nsec3 hash breaking. In *Network Computing and Applications (NCA), 2014 IEEE 13th International Symposium on*, pages 137–144. IEEE, 2014.

[46] F. Weimer. DSA-1571-1 openssl - Predictable Random Number Generator, 2008.

[47] H. Yang, E. Osterweil, D. Massey, S. Lu, and L. Zhang. Deploying cryptography in internet-scale systems: A case study on dnssec. *Dependable and Secure Computing, IEEE Transactions on*, 8(5):656–669, 2011.

[48] Z. Zorz. Lenovo.com Hijacking Made Possible by Compromise of Webnic Registrar, 2015.