Automating Trusted Key Rollover in DNSSEC

Gilles Guette
IRISA/Université de Rennes 1
Campus de Beaulieu
35042 Rennes CEDEX, FRANCE
gilles.guette@irisa.fr

September 8, 2008

Abstract

The Domain Name System (DNS) is a distributed tree-based database largely used to translate a human readable machine name into an IP address. The DNS security extensions (DNSSEC) has been designed to protect the DNS protocol using public key cryptography and digital signatures. Every secure DNS zone owns at least a key pair (public/private) to provide two security services: data integrity and authentication. To trust some DNS data, a DNS client has to verify the signature of this data with the right zone key. This verification is based on the establishment of a chain of trust. To build this chain of trust, a DNSSEC client needs a secure entry point: a zone key configured as trusted in the client. In this paper, we study the management problem of this kind of key also call the trusted key rollover problem. We propose a new resource record (RR) to automate this rollover and avoid the inconsistency problem between the resolver key set and the name server key set. Without our new record and solution, this problem needs an administrator action to be solved.

Keywords: DNSSEC, network security, key management, key rollover.

1 Introduction

The Domain Name System (DNS) [1] is a hierarchical distributed database mostly used to translate host names into IP addresses. The DNS protocol does not include any security services such as data integrity and authentication. This lets some vulnerabilities in the protocol [6, 16, 5], that is why the Internet Engineering Task Force (IETF) has developed the DNS security extensions (DNSSEC).

DNSSEC [8, 11, 2, 4, 3] uses public-key cryptography to provide DNS data integrity and authentication. Each node of the DNS tree, called a zone, owns at least a key pair used to generate digital signatures of the DNS zone information. The basic data unit of this information is a resource record (RR). Each RR has a particular type that indicates which kind of data it contains. For example, a DNSKEY RR contains a zone key, a RRSIG RR contains a signature and an A RR contains an IPv4 address.

In order to trust DNS data, a resolver (the DNS client) builds a chain of trust [9] by walking through the DNS tree from a secure entry point (*i.e.*, a trusted key statically configured in the resolver, typically a top level zone) to the queried resource record. A resolver is able to build a chain of trust if it owns a secure entry point and if there are only secure zones on the path from the secure entry point to the queried zone.

Existing key rollover mechanisms only updates keys on the name servers, resolvers that have configured the old keys as trusted are not notified that these keys are being replaced. Consequently, the static key configuration in a resolver raises consistency problems between keys deployed in a zone and trusted keys configured in a resolver. If all keys statically configured in a resolver becomes out of date, this resolver cannot perform secure name resolution any longer. A manual intervention is required by the administrator to replace the resolver key set. It is obvious that an administrator that administrate a lot of machine will become quickly overload or will never deploy DNSSEC on its machines. This emphasizes the need of the mechanism to totally automate the key rollover on the client side that we define in this paper.

In Section 2, we present the DNSSEC architecture and the secure name resolution process. Then, we describe in Section 3, three methods of automated key rollover. We show in Section 3.4 that all these methods have some limitations and we present in Section 4 our proposition to overcome these limitations.

2 DNSSEC architecture and secure name resolution

The DNS is a distributed tree-based database. In this Section, we describe the architecture of DNSSEC and the interactions between the components.

2.1 Domains and zones

A domain is a subtree of the DNS tree. The name of a domain is the concatenation of all the node's label from the root of the subtree to the root of the DNS tree. Since two nodes having the same parent node have a different label, unicity of domain names is ensured. A domain can be included in another domain, for example the `irisa.fr.` domain is included in the `fr.` domain, as shown on Figure 1.

Each domain is constituted of one or several zones. The zone is the administrative unit of the DNS and is represented by a node in the DNS tree. A zone is managed by one or several name servers that store the zone information in a zone file. This zone file contains all the zone resource records.

2.2 DNS entities

Three entities with distinct roles are present in the DNS architecture: the authoritative name server, the cache server and the resolver.

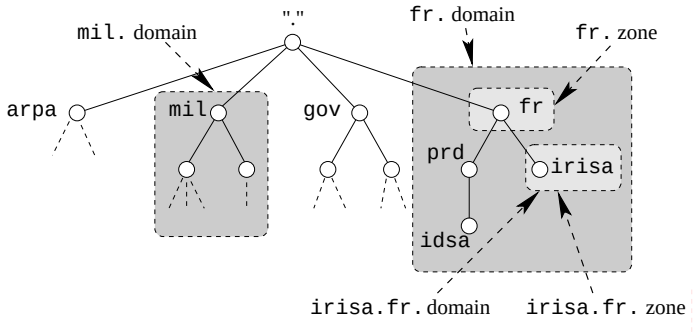

Figure 1: DNS domains and DNS zones.

2.2.1 The authoritative name server

The name server is authoritative on a DNS zone. It stores resource records in its zone file. Every resource record is associated with a DNS name. The name server receives DNS queries about a DNS name and replies with the corresponding resource records contained in its zone file.

2.2.2 The cache server

A cache server is never authoritative on a zone. It is generally located on a local network to receive queries from local resolvers. A cache server sends responses to these queries using resource records that have been previously received. If the cache server has not stored the answer for a particular query, it forwards the query to the most appropriated authoritative server it knows. Then, upon receiving the DNS response, it keeps a copy in memory and sends it to the resolver that had previously initiated the name resolution. Using cache servers allows to minimize the burden on authoritative servers [17, 13, 7].

2.2.3 The resolver

The resolver is the local entity that receives requests from applications and sends DNS queries to name servers or cache servers. After having performed the name resolution, the resolver returns the answer to the application.

2.3 DNSSEC chain of trust

DNSSEC uses public key cryptography and defines new resource records to store keys and signatures. Each secure zone owns one or several zone keys. The public part of each key is stored in a DNSKEY resource record. The private part of a zone key is kept secret and is often stored in a secure location. This private part is used to generate a digital signature of each resource record in the zone file. Then, each signature is stored in a RRSIG resource record. To trust a resource record, a resolver must verify at least one signature of this resource record with a trusted zone key.

2.3.1 Two types of DNSSEC keys

There are two ways for a resolver to trust a zone key. Either this key is *trusted key* that is configured in the resolver or the resolver trusts a Delegation Signer (DS) RR [11] that authenticates this key. A DS RR is a resource record stored in the parent zone that authenticates a key of the child zone. Basically, a DS RR contains a key identifier and a hash of a child zone public key. The DS RR is signed by parent zone keys.

A DS resource record creates a secure link between a parent zone and its child zone, together with a dependency. Indeed, when a key is renewed in the child zone, the DS RR must be renewed in the parent zone. This implies a communication between the child zone and the parent zone. To minimize the burden due to this additional data exchange, some keys do not have an associated DS RR. The DS model introduces a distinction between two types of keys: the Key Signing Keys (KSK) and the Zone Signing Keys (ZSK). A KSK has an associated DS RR in the parent zone and signs only DNSKEY resource records. A ZSK does not have any associated DS RR and signs all the resource records in the zone file.

Even though DS identifies two roles for keys, KSK and ZSK, there is no requirement for zones to use two different keys for these roles. It is expected that many small zones will only use one key, while larger zones will more likely use multiple keys [11].

2.3.2 DNSSEC signature verification process

To decrease the size of the zone file, resource record associated to the same name and having the same type are grouped and signed together. For example, if a zone owns three DNSKEY RRs, these three RRs are grouped in a DNSKEY RRset (or keyset). Then, each key generates a signature for this DNSKEY RRset. Hence, we have three signatures associated to the DNSKEY RRset. The verification of only one signature is sufficient to trust the DNSKEY RRset and hence the three keys it contains.

During a secure name resolution, a resolver builds a chain of trust, that is to say it starts from a trusted key and follows the path to the queried resource records while verifying signatures of resource records and secure link represented by DS-DNSKEY records.

Figure 2 shows the different steps a DNSSEC client having only the root zone configured as secure entry point (SEP) follows to trust a resource record of the `irisa.fr.` zone.

At each step, we have the same scheme: a DS resource record allows to trust a KSK, a KSK allows to trust all the zone key (KSK and ZSK) and a ZSK allows to trust other resource records of the zone.

In the next section, we present a brief overview of existing methods for automated trusted key rollover.

3 Related work

During the last two years, three methods [12, 18, 10] have been designed to automatically update the trusted key set of a resolver. In this section, we describe briefly these three methods and we discuss their limitations.

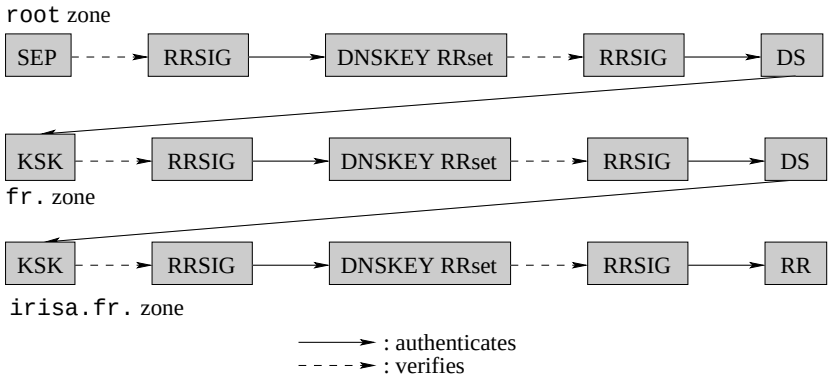

Figure 2: Establishment of a chain of trust.

3.1 The threshold-based algorithm

The automated trusted key rollover method presented in [12] compares the key-set received in a DNSSEC answer with the trusted keys owned by the resolver. This comparison is done according to two parameters.

3.1.1 The M and N parameters

To accept a new key, the resolver's administrator must set two parameters M and N. The M criterion is the minimum number of configured trusted keys that should directly verify RRSIGs over the new DNSKEY RRset.

The N criterion is the number of differences between configured trusted keys having the SEP bit set [15] and received zone keys having the SEP bit set. The SEP bit is the seventh bit of the flags field of a DNSKEY RR. This bit indicates a KSK and hence a key that can be used as a secure entry point. Note that $N \geq 1$.

The M and N criteria set by the resolver's administrator define the acceptance policy for a new trusted key.

3.1.2 The threshold-based algorithm

Algorithm 1 describes the threshold-based rollover method [12].

if the DNSKEY RRset in the zone has been updated **then**
  if at least M configured trusted keys directly verify the related RRSIGs over the new DNSKEY RRset **then**
    if the number of configured trusted keys that verify the related RRSIGs over the new DNSKEY RRset $>$ N **then**
      all the trusted keys for the particular secure entry point are replaced by the set of keys from the zone DNSKEY RRset that have the SEP flag set.
    end if
  end if
end if

Algorithm 1: : The threshold-based algorithm

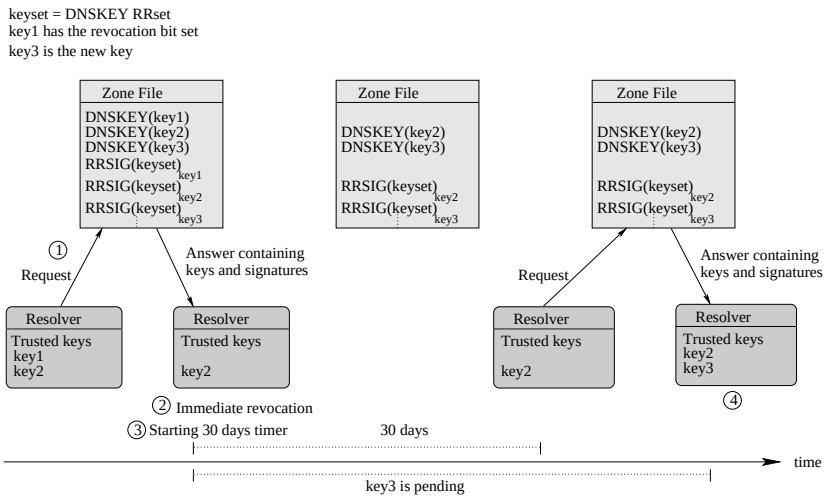

Figure 3: Rollover method with 30 days timer.

3.2 Update timers method

A second automated trusted key rollover method has been proposed in [18]. This method defines a revocation bit in the Flags field of the DNSKEY resource record. When this bit is set, it means that the key must be removed from the resolver's trusted key set. This revocation is immediate. Figure 3 shows the principles of this method.

Firstly, the resolver sends a query about a DNS name (see ①) and receives a DNSKEY RRset. If one of these RRs has the revocation bit set, the revocation is immediate and the resolver updates its trusted key set (see ②). To mitigate compromised key attacks, the resolver starts a 30 days timer and keeps a trace of all the keys that signed the DNSKEY RRset (see ③). If the resolver receives a response containing a DNSKEY RRset without the new keys but validly signed before the expiration of the timer, the resolver considers this information as a proof that something goes wrong (attack or misconfiguration). Consequently, it stops the acceptance of the new key and resets the timer.

In the same way, if all the keys that have signed the DNSKEY RRset are revoked before the timer expiration, the resolver stops the acceptance of the new key. Once the timer expires, the next time the resolver receives a valid DNSKEY RRset containing the new key it accepts this key as trusted (see ④).

3.3 The automated trusted key rollover algorithm

In [10], the authors propose to use the first of the reserved bits of the Flag field and to call it the *under changes* bit. That gives the following meaning to this bit: when this bit is set the key contained in the DNSKEY RR is under changes and is going to be removed from the DNS zone file.

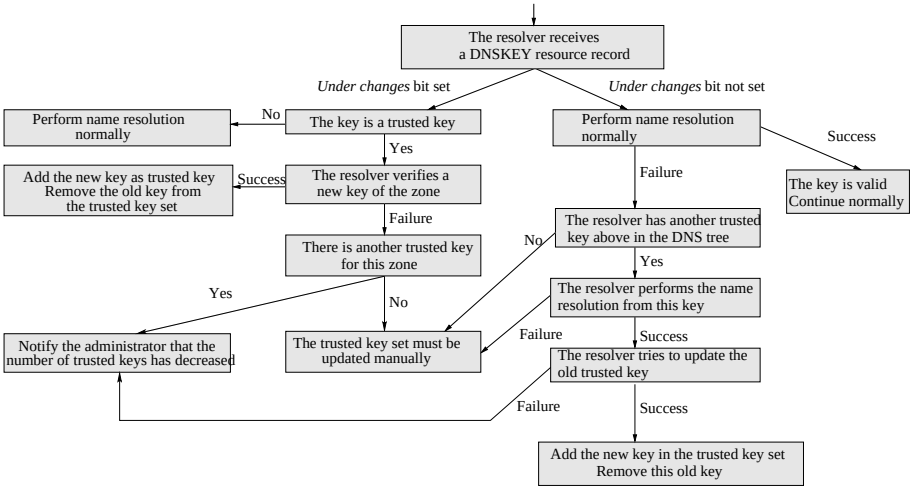

Figure 4: Resolver's behavior.

3.3.1 Description

When a zone decides to replace one of its keys, it sets the *under changes* bit in the DNSKEY RR containing this key. Advices for the duration of the rollover period can be found in [14].

Figure 4 shows the resolver behavior when it receives a DNSKEY RR. Either the *under changes* bit is set, so the resolver checks if this key is configured as trusted. If this is not the case, this key rollover will have no impact on the resolver behavior and the resolver must continue normally the current name resolution. If the received key is a trusted key for the resolver, it tries to validate another key of the zone. To validate a new key, the resolver must verify k signatures of the DNSKEY RRset, where k is a threshold set by the resolver's administrator. When a zone has n keys, this threshold allows to resist to $n - k$ compromised keys. If it succeeds, the old key is removed and the new key is added in the resolver configuration file. If the new key validation fails, the resolver raises a warning and the change in the resolver's configuration file must be made manually by the administrator.

On some resolvers, requests on a given zone could be largely spaced out, more than one month for example. Consequently, a resolver cannot perform name resolution on a zone during the rollover period of a trusted key. This implies that the trusted key configured in such a resolver may have been renewed and removed from the zone file during this period of *resolver inactivity*.

This can raise two critical events if the *under changes* bit is not set: the resolver has another trusted key belonging to an ancestor's zone of the zone that has changed its zone key, or the resolver has no such trusted key.

If the *under changes* bit is not set the resolver performs name resolution normally. If the name resolution fails the resolver must try to update its trusted keys using another key for the name resolution. If the resolver cannot perform a DNSSEC resolution either because it has no valid trusted key or because the resolution is not secure (some data are retrieved from unsecure DNS zones) the

resolver cannot update its configuration file and the trusted key update must be made manually.

The algorithm depicted on Figure 4 shows that in some cases a manual administrative action is needed. This is due to possible misconfiguration in DNS zones or to the possibility for a resolver to have sent no query to a zone during the rollover period of the zone, and therefore to have received no notification of key changes.

3.4 Automated trusted key rollover limitations

The three key rollover methods described above have some limitations. The major problem with the second method [18] is that it introduces extra delay (30 days) and requires a lot of management for the zone administrator. Indeed this method implies a lot of constraints about the minimal number of keys in a zone, the number of trusted keys and the rollover frequency due to the 30 days timer. When a server creates a new key, this key is pending as defined in [18] and cannot be used during 30 days after its creation. This implies that if a zone owns n keys and a key is replaced every $\frac{30}{n-1}$ days, no resolvers will be able to update their trusted key set for this zone. All the keys will be revoked prior to the timer expiration.

The two other methods do not have time constraints, but do not force resolvers to query servers. The same problem with spaced queries may arise and a resolver will be unable to update its trusted key set.

We propose in the next section a new resource record to solve the limitations of the previous method and to allow automated rollover in optimal conditions.

4 A new resource record to automate the rollover

As for other applications using cryptography, key rollover in DNSSEC is a critical mechanism. In DNSSEC, the key rollover is done on the server side, but the client must be notified that a given key has been replaced on a given zone. Some information must be included in the zone files. They allow a resolver to be aware of the interrogation period of a zone and hence to be able to automatically update its trusted key set. To do so, three pieces of information are needed:

- the key rollover period of the zone,
- the number of zone keys,
- the maximum number of keys replaced at a time.

Problems with automated key rollover in DNSSEC are due to the current configuration of the DNS. The keys are present on name servers and copy of these keys are present in cache servers and resolvers but no mechanism exists to keep the consistency of this redundant information.

It is necessary to check periodically if the trusted keys that are present on a resolver are up to date. If the resolver do not query name servers periodically to know if its keys are up to date, they may become useless. This is the case if the resolver does not query a zone during a given time (*e.g.*, accessing a machine for software update is typically done with low frequency). To notify resolvers

of key renewal in a zone, we design a new resource record named Key Renewal Information (KRI) RR.

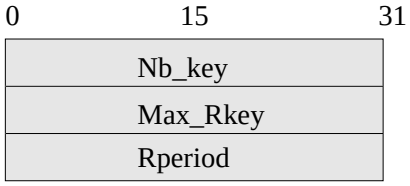

Figure 5: The KRI RR wire format.

The KRI RR (see Figure 5) contains information about the server's key management policy. These three fields are needed to calculate the interrogation period of a given zone that a resolver should not exceed to automatically update its trusted key set.

The KRI RR is composed of three fields of four bytes each: *Nb_ key*, *Max_ Rkey* and *Rperiod*. The *Nb_key* field is the number of zone keys in the zone file. The *Max_ Rkey* field is the maximum number of keys a zone can replace at a time. The *Rperiod* field is the minimal rollover period of the zone (in seconds). A length of 32 bits is needed for the *Nb_ key* and *Max_ Rkey* because a zone can own up to 2^{16} keys by cryptographic algorithm defined in the DNSSEC standard (the associated algorithm is specified in the *algorithm* field of the DNSKEY RR). We choose a length of 32 bits for these fields because they must be larger than 16 bits and 4 bytes words are easier to managed. This let also enough flexibility in the evolution of the number of cryptographic algorithm that could be used in DNSSEC. The *Rperiod* is a 32-bit integer allowing to define a rollover period greater than one century, which is largely sufficient. These three fields are sufficient for a resolver to easily calculate the interrogation period of a zone to avoid the critical cases we have described in the above sections. We will give the formula in the following. The total length of 12 bytes for this record is not a problem. Just for comparison purpose, remember that the size of a key generally used in DNSSEC and stored in a DNSKEY RR is 1024 bits (128 bytes).

In the next subsections, we define and prove the interrogation period formula. This formula can be used by all the three key rollover methods presented before. The first formula does not take into account recursive cache servers. We enhanced this formula in the following.

4.1 Interrogation period without recursive cache servers

In this subsection, we present the first formula to calculate the interrogation period without taking into account the presence of cache servers. Thus, this formula does not take into account records that may be in cache during TTL time and potentially give inconsistency (old DS RRs and new DNSKEY RRs for example).

Let Nb_key be the number of zone key, Max_Rkey be the maximum number of keys replaced at a time and *Rperiod* be the rollover period of the zone.

Let R be a resolver. We denote by $S_{(R)}$ the minimum number of DNSKEY

RR signatures R must positively verify to accept a new key. We have $S_{(R)} \geq 1$ because the RFC 2535 [8] specifies that to validate a RR at least one signature must be verified. Let *Iperiod* be the maximum interrogation period. *Iperiod* is the time upper bound of a possible automated key rollover. If *Iperiod* is exceeded, manual intervention is required. Indeed, we ensure that after *Iperiod*, there will be no common key between the key set of the name server and the key set of the resolver R.

We have some constraints on the parameters:

Theorem 1. $\forall R, S_{(R)} \leq Nb\_key - Max\_Rkey$.

Proof. After a key rollover, there are in the zone file at least $Nb\_key - Max\_Rkey$ signatures of the DNSKEY RRset generated by old keys. To accept a new key, this number of signatures must be greater than or equal to $S_{(R)}$. □

Corollary 1. $Nb\_key \geq Max\_Rkey + 1$.

Proof. From theorem 1 and $S_{(R)} \geq 1$. □

Theorem 2. $S_{(R)} \in [1, .., Nb\_key - 1]$.

Proof. During a key rollover at most $Max\_Rkey \geq 1$ keys are replaced, theorem 1 gives $S_{(R)} \leq Nb\_key - 1$. □

We can now define the interrogation period formula (*Iperiod*).

Theorem 3. $\forall R, Iperiod = \lfloor \frac{Nb\_key - S_{(R)}}{Max\_Rkey} \rfloor \times Rperiod$

Proof. Let $n \in \mathbb{N}^*$ be the maximum number of key rollovers such as $Nb\_key - n \times Max\_Rkey \geq S_{(R)}$.

Then $Nb\_key - n \times Max\_Rkey \geq S_{(R)} \Leftrightarrow n \times Max\_Rkey \leq Nb\_key - S_{(R)}$.

$\Leftrightarrow n \leq \frac{Nb\_key - S_{(R)}}{Max\_Rkey}$

As $n \in \mathbb{N}^*$ and by definition of n,

$n = \lfloor \frac{Nb\_key - S_{(R)}}{Max\_Rkey} \rfloor$.

Therefore $Iperiod = x \times Rperiod$, where x is the largest integer such as $Nb\_key - x \times Max\_Rkey \geq S_{(R)}$

Thus $x = n$ and $Iperiod = \lfloor \frac{Nb\_key - S_{(R)}}{Max\_Rkey} \rfloor \times Rperiod$. □

Figure 6 shows a use case of the previous formula.

4.2 Interrogation period formula taking into account recursive cache servers

One important thing impacting on the efficiency of the automated key rollover is the TTL time during which resource records are kept in a recursive cache server. The cache servers are the reason why DNS scales so well, nevertheless during TTL time there may be inconsistencies between records in the cache server and records in the zone file, as shown on Figure 7. That is why we must consider cache server in our formula.

The scenario in Figure 7 shows clearly how inconsistencies appear:

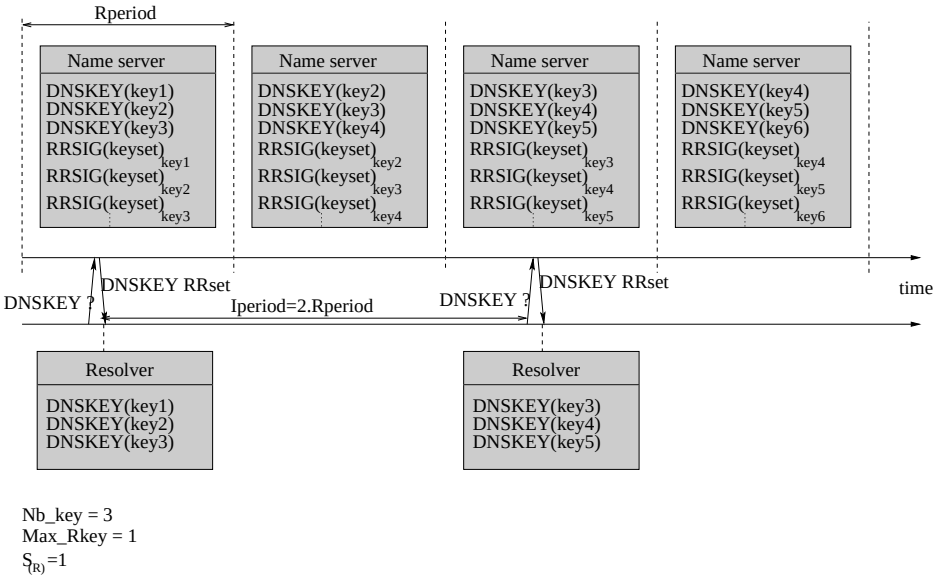

Figure 6: Interrogation period calculation formula.

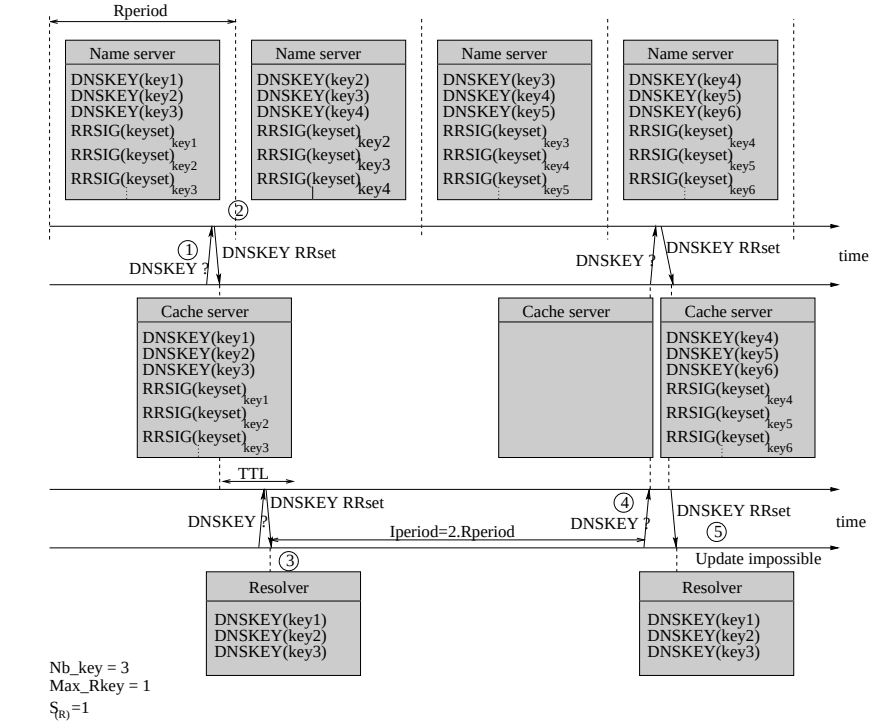

Figure 7: Trusted key set update problem.

- A cache server sends a query to a name server and keeps the answer during TTL time ①.
- Just after sending the answer, the name server replaced some DNSKEY RRs. There are inconsistencies between the name server and the cache server ②.
- A resolver R sends a query to its cache server and receives the old keys ③. Then, it waits at most $\lfloor \frac{Nb\_key - S_{(R)}}{Max\_Rkey} \rfloor \times Rperiod$ (Theorem 3), until its next query on this zone. Here, we have $Iperiod = \lfloor \frac{3-1}{1} \rfloor \times Rperiod = 2.Rperiod$.
- R sends another query to its cache server. This cache server does not know the answer so it forwards the query to the appropriate name server and receives an answer containing key4, key5 and key6. The cache sends back the answer to the resolver R ④.
- R cannot update its trusted key because $|\{\text{key1, key2, key3}\} \cap \{\text{key4, key5, key6}\}|=0 < S_{(R)}$. Due to the old keys kept in the cache server, the $Iperiod$ has been one $Rperiod$ time too long ⑤.

So, we must ensure that when a resolver obtains keys from a cache server, the automated keys update remains possible. The next theorem takes into account the presence of cache servers to calculate the interrogation period.

We assume that $Rperiod \geq 2 \times TTL$. For information, recommendations included in the standard suggest that a ZSK is changed every week and a KSK is changed every month. Moreover, the TTL of a record is generally positioned at one day per default.

We use the notation with an open bracket: $[\ldots[$, to specify an open upper bound.

Theorem 4. $\forall R, Iperiod = \lfloor \frac{Nb\_key - S_{(R)}}{Max\_Rkey} \rfloor \times Rperiod - TTL$.

Proof. We note $n = \lfloor \frac{Nb\_key - S_{(R)}}{Max\_Rkey} \rfloor$. Let t_1 be the time when a resolver R sends a query to a given zone. We have:

$\exists a \in \mathbb{N}$, such as $a \times Rperiod \leq t_1 < (a+1) \times Rperiod$. We can define $t2$ by $t_2 = t_1 + n \times Rperiod - TTL$.

Now we show that t_2 is the latest possible time for the next interrogation we want to have when there is no more than n rollovers between $t1$ and $t2$.

- If $t_1 \in [a \times Rperiod; a \times Rperiod + TTL[$, data present in cache servers at t_2 is from $[(a+n-1) \times Rperiod; (a+n) \times Rperiod[$
  because $[(a+n) \times Rperiod - TTL; (a+n) \times Rperiod[\subset [(a+n-1) \times Rperiod; (a+n) \times Rperiod[$.
  - If data present in cache at t_1 is from $[(a-1) \times Rperiod; a \times Rperiod[$ then there is at most n key rollovers between t_1 and t_2.
    $\Rightarrow R$ can update its trusted keys.
  - If data present in cache server at t_1 is from $[a \times Rperiod; (a+1) \times Rperiod[$ then there is at most $(n-1)$ key rollovers between t_1 and t_2.
    $\Rightarrow R$ can update its trusted keys.

- If $t_1 \in [a \times Rperiod + TTL; (a+1) \times Rperiod[$, data present in cache server at t_1 is from $[a \times Rperiod; (a+1) \times Rperiod[$.
  - If data present in cache server at t_2 is from $[(a+n-1) \times Rperiod; (a+n) \times Rperiod[$ then there is at most $(n-1)$ key rollovers between t_1 and t_2.
    $\Rightarrow R$ can update its trusted keys.
  - If data present in cache server at t_2 is from $[(a+n) \times Rperiod; (a+n+1) \times Rperiod[$ then there is at most n key rollovers between t_1 and t_2.
    $\Rightarrow R$ can update its trusted keys.

So, $Iperiod = t_2 - t_1 \geq n \times Rperiod - TTL$.

Let us suppose by contradiction that $t_2 > t_1 + n \times Rperiod - TTL$ and let t_1 be $\max[a \times Rperiod; a \times Rperiod + TTL[$. t_1 exists because the interval is finite (time in seconds).

Then $t_2 \geq t1 + n \times Rperiod - TTL \Leftrightarrow t_2 \geq a \times Rperiod + TTL + n \times Rperiod - TTL$

$\Leftrightarrow t_2 \geq (a+n) \times Rperiod$

$\Leftrightarrow t_2 \in [(a+n) \times Rperiod; \infty[$

If data present in cache servers at t_1 is from $[(a-1) \times Rperiod; a \times Rperiod[$ then, there are at most $(n+1)$ key rollovers between t_1 and t_2. If there has been $(n+1)$ key rollover, R cannot update its trusted keys.

Hence, $t_2 \leq t_1 + n \times Rperiod - TTL$ and $Iperiod = t_2 - t_1 \leq n \times Rperiod - TTL$. So, $Iperiod = n \times Rperiod - TTL$. □

We can notice that we can easily obtain the TTL value of a zone. Indeed, every resource record has an associated TTL value sent in every DNS message.

4.3 Round Trip Time and impact on interrogation period

Another thing that can impact on the interrogation period is the round trip time (RTT). Two critical cases can arise and are described below:

1.
   - The resolver sends a query to a given zone using the previous formula. The answer containing keys is sent just before a key rollover in the zone.
   - With the RTT, the query arrives just after the rollover.
   - In this case, the threshold $S_{(R)}$ cannot be reached and the resolver cannot update its trusted keys set.

2.
   - A name server sends a response and then replace some zone keys.
   - Because of the RTT the response arrives at the resolver after the key rollover.
   - Inconsistencies exist during $(TTL + RTT/2)$ seconds.
   - The resolver must take into account the RTT to be able to update its trusted keys set.

DNSSEC messages are sent over the UDP protocol, so some packet may be lost. When a resolver sends a query, this query or the associated answer may never arrive. In this case the resolver re-sends the query to mitigate the risk of a packet loss.

Re-sending queries or congestion of network links may introduce additional delays. The threshold implemented in the resolver to support this delay are variable but basically if the delay exceeds the timer t_R (known but depending on the resolver implementation), the packet is considered as lost. We can use t_R as an upper bound for the RTT.

Theorem 5. $\forall R, Iperiod = \lfloor \frac{Nb\_key - S_{(R)}}{Max\_Rkey} \rfloor \times Rperiod - TTL - t_R$.

Proof. Same proof as for Theorem 4 with $t_2 = t_1 + n \times Rperiod - TTL - t_R$. □

The formula of the Theorem 5 takes into account cache server and RTT. This formula allows a resolver to automatically update its trusted key set in every cases.

4.4 KRI resource record management

The KRI RR is present in the zone file of a zone. Hence, a resolver having configured some trusted keys for a given zone must have received its KRI RR to avoid the critical cases described in this paper. With data contained in the KRI record and with local policy parameters, a resolver can calculate the interrogation period of a given zone.

Information contained in the KRI RR is placed by the administrator of the zone. This administrator may want to change its key rollover policy and information in its KRI RR. With the new information contained in the KRI RR the server's administrator calculates: $\lfloor \frac{Nb\_key-1}{Max\_Rkey} \rfloor \times Rperiod - TTL$. If the result is greater than the one obtained with the previous information, then the new key rollover policy should be applied immediately.

If the result is smaller than the one obtained with the previous information the administrator must update parameters in the KRI RR and must wait for $\lfloor \frac{Nb\_key-1}{Max\_Rkey} \rfloor \times Rperiod - TTL$ time, before to apply its new key rollover policy. This is done in order not to disturb the automated trusted key rollover.

This process allows all the resolvers to obtain the new parameters and to calculate the new interrogation period (see Theorem 5) even if the new interrogation period is smaller than the previous one. After this delay $\lfloor \frac{Nb\_key-1}{Max\_Rkey} \rfloor \times Rperiod - TTL$, the new key rollover policy is effective in the zone.

5 Conclusion and future work

The key rollover is a critical mechanism for all applications using public key cryptography. Nevertheless, in DNSSEC the key rollover raises some consistency problems between keys in name servers and their copies in cache servers and resolvers. In the case of a DNSSEC resolver, if the key set of the resolver becomes out of date it will be unable to accept new keys as trusted and hence will be unable to perform any name resolution. For the common user, as it does not know specific IP addresses, this means no access to any web server for

example. Moreover, to update the resolver's key set an administrator action is needed. In the current standard of DNSSEC, there is no automated key rollover mechanism working in any scenario. It would be of benefit for the deployment of DNSSEC to have an automated rollover method allowing to avoid the work overload of the administrator.

In this paper, we have designed a new resource record and presented its management. This record contains the required information to automate the trusted key rollover. We have defined and prove the formulas allowing a resolver to know the interrogation period of a given zone to enable the automated trusted key rollover. Our formulas take into account recursive cache servers, RTTs and possible packet losses during a transaction. With this new record, the update of the trusted key set of a resolver can be automatically done.

The next steps of this work is to simulate the DNS tree with frequent changes in the zone key rollover policies, to assess the efficiency of our formula under an heavy workload.

References

[1] P. Albitz and C. Liu. *DNS and BIND.* O'Reilly & Associates, Inc., Sebastopol, Californie, 4th edition, January 2002.

[2] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. RFC 4033, March 2005.

[3] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. RFC 4035, March 2005.

[4] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. RFC 4034, March 2005.

[5] D. Atkins and R. Austein. Threat Analysis of the Domain Name System. RFC 3833, August 2004.

[6] S. M. Bellovin. Using the Domain Name System for System Break-Ins. In *Proceedings of the 5th Usenix UNIX Security Symposium*, pages 199–208, June 1995.

[7] E. Cohen and H. Kaplan. Proactive Caching of DNS Records: Addressing a Performance Bottleneck. In *Symposium on Applications and the Internet*, pages 85–94, January 2001.

[8] D. Eastlake. Domain Name System Security Extensions. RFC 2535, March 1999.

[9] R. Gieben. Chain of trust: The parent-child and keyholder-keysigner relations and their communication in dnssec. Master's thesis, University of Nijmegen, 2001.

[10] G. Guette, B. Cousin, and D. Fort. Algorithm for DNSSEC Trusted Key Rollover. In *The International Conference on Information Networking (ICOIN)*, volume 3391 of *LNCS*, pages 679–688. Springer, January 2005.

[11] O. Gundmundsson. Delegation Signer Resource Record. RFC 3658, December 2003.

[12] J. Ihren, O. Kolkman, and B. Manning. An In-Band Rollover Algorithm and an Out-Of-Band Priming Method for DNS Trust Anchors. Draft IETF, work in progress, October 2005.

[13] J. Jung, E. Sit, H. Balakrishan, and R. Morris. DNS Performance and the Effectiveness of Caching. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop '01*, pages 153–167, November 2001.

[14] O. Kolkman and R. Gieben. DNSSEC Operational Practices. RFC 4641, September 2006.

[15] O. Kolkman, J. Schlyter, and E. Lewis. Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag. RFC 3757, April 2004.

[16] C. L. Schuba. Addressing Weaknesses in the Domain Name System. Master's thesis, Purdue University, Department of Computer Sciences, August 1993.

[17] E. Sit. A Study of Caching in the Internet Domain Name System. Master's Thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Sciences, May 2000.

[18] M. StJohns. Automated Updates of DNSSEC Trust Anchors. Draft IETF, work in progress, January 2006.