

## KRO: A KEY ROLLOVER ALGORITHM FOR DNSSEC

Gilles Guette [gilles.guette@irisa.fr](mailto:gilles.guette@irisa.fr)  
*Irisa/INRIA, Campus de beaulieu France.*

Olivier Courtay [olivier.courtay@enst-bretagne.fr](mailto:olivier.courtay@enst-bretagne.fr)  
*ENST-Bretagne, 2 rue de la Châtaigneraie, France.*

**Abstract:** An IP address can be obtained from a machine name through a distributed database called Domain Name System (DNS). The DNS is used by every computer connected to Internet but there is no security service included to protect DNS data transfers. DNS has therefore evolved into a secure version: DNSsec. DNSsec uses public-key cryptography and needs to manage the keys. The key rollover is a complex mechanism that suffers from a lack of automation. In this paper, we present an algorithm that automates the key rollover process in DNSsec and we start a discussion on several hot topics.

**Keywords:** DNSsec, Chain of trust, Authentication, Integrity, Key management.

### 1. INTRODUCTION

In order to communicate over the Internet, two computers need to know their respective IP addresses. From a machine, name an IP address can be obtained through a distributed database called Domain Name System (DNS) [7,8]. This service translates a human-readable computer name into the associated IP address. DNS may also give other information about computers such as their name server, the type of their processor, etc. All this information are stored inside the zone file in Resource Records. The original design of DNS does not include any security services able to protect DNS transactions or DNS data. The DNS stays vulnerable to numerous attacks such as packet forging or modification, cache pollution or server impersonation, etc.[4,1].

To solve this lack of security, the DNS protocol has been extended to a secure version: DNSsec [2,6].

DNSsec gives two essential security services for the Domain Name System: it guarantees data integrity and authentication. In order to effectively provide these services, DNSsec uses public-key cryptography and digital signatures. Specific Resource Records (RR) were added to the DNS to manage keys and necessary signatures for public-key cryptography. These new records are KEY, SIG, NXT [2] and DS [6]. In DNSsec, every resource record is signed (except in one specific case for glue NS at the delegation point) and this signature is stored in a SIG resource record.

DNSsec gives also an additional workload to DNS administrators. New tasks will include the key management (creation and renewal), the signature of the zone and the notification to the server of the parent zone that some records have changed, in order to create the right DS record.

In this paper, we present an algorithm that automates the key rollover process in DNSsec. In the first part, we discuss about key management mechanisms in DNSsec and about the need of management automation. In the second part, we present our algorithm for key rollover, its principles and its advantages. Then, we talk about some advanced problems.

## **2. KEY MANAGEMENT IN DNSSEC**

Information contained in DNS zone files are public, confidentiality is therefore not required. This data is not cyphered and DNS data traffic travels through the network without any protection. Nevertheless, integrity and authentication are required. DNSsec provides securisation of the DNS thaks to data integrity and authentication. DNSsec is based on the following two points:

- Securing each zone.
- The trust of the zone information.

### **2.1. DNSsec Chain of Trust**

In each zone file are present SIG and KEY resource records (RR). SIG RR contain signature of associated resource records and KEY RR contain the keys used for the creation of signatures. Two kinds of key exist: Key Signing Key (KSK) and Zone Signing Key (ZSK) [6]. KSKs are used to sign only the KEY RRset (the set of keys present in the zone file) and ZSKs are used to sign every resource record in the zone file.

Thus, we have in a zone file some resource records secured by digital signature providing their integrity. For each KEY resource record there are as many associated SIG resource records as existing keys and for the others types of records there are as many associated SIG resource records as existing ZSK. The critical point becomes the key used to create these signatures, because it is necessary to trust it.

To validate a KEY, or in other terms to trust it, DNSsec uses the DNS-tree model to establish a chain of trust [5] beginning from the root of the DNS tree (which is trusted) or from a secure entry point to the interrogated zone. To create this chain, a verifiable relation between child zone and parent zone must exist: this is the role of the DS resource record [6]. This record contains some information allowing the authentication of one KSK of the child zone. To trust the root key or a secure entry point key, the key must preably be statically configured in a local file as a trusted key.

To validate the chain of trust, DS specifies that at least one valid DS resource record for the KSK of the child zone must be stored in the parent zone. This implies that when a zone decides to change its KSK, it must contact its parent zone and send its new KSK to notify that some changes have occurred. Then the parent zone can create the correspondant DS resource record or delete a DS resource record which is becoming obsolete.

## 2.2. Through an Automated Rollover

The keys which are used in DNSsec need to be changed periodically because from compromission point of view a key wears. The more a key is used to cypher data, the more it weakens. No automated mechanisms exists to do a key rollover between DNS entities. Currently, this rollover is made by an agreement between the administrators of the two zones using any secure communication mean outside DNS control (with a signed cyphered mail for example). When the new key is sent with such a mechanism, the integrity of sent data is not under the control of the DNSsec. Moreover such a mechanism is in contradiction with the deployment of an entirely automated solution. Such a mechanism does not provide any synchronization between the parent zone and its child zone. Thus, even if the parent zone is quickly informed that changes have occurred in its child zone, due to (re)signature policy which could introduce a large delay, the changes concerning the child zone (DS record) will be effective later, only after re-signature of the parent zone. This can result into consistency problems and thus prevent the check of the chain of trust as we will see in the following scenario:

1. The child zone changes its KSK.

2. The child zone administrator informs the administrator of its parent zone that changes have been made.
3. The child zone administrator deletes the old KSK before the parent zone is resigned.
4. The child zone resource records are no longer verifiable: there is no DS resource record corresponding to the KSK. The chain of trust is broken.

In the above scenario, a data consistency problem occurs which makes the child zone not-secured from an external point of view, like all the zones located below the child zone. Indeed, as soon as the chain of trust is broken, no information located below the break point is trustworthy because this information is not verifiable. Moreover, for a large zone such as the ".com" zone, a non automated rollover is impossible. Assuming that only 100,000 child zones of the ".com" zone are secured and scheduled a KSK change only once a year would result in approximately 300 changes per day for the ".com" zone administrator. That is why it is necessary to automate the key rollover process.

### **3. KRO, A KEY ROLLOVER ALGORITHM**

In this section, we present an algorithm allowing the automation of the key rollover and the re-signature of a zone. We make the assumption that the child zone is already a secured zone. We will discuss later the case where the zone is not (yet) a secured zone. During the change of keys, two cases can occur:

- the child zone changes its ZSK.
- the child zone changes its KSK.

If the change concerns a ZSK, all actions are under the responsibility of the zone which accomplishes the change. Indeed, the only operations are the creation or the deletion of a ZSK and the re-signature of the zone. The KSKs signing the Key RRset are not modified, it is not necessary to modify the DS record located in the parent zone. It is the simplest case of rollover because there is no need to involve the hierarchy relation.

If the change concerns a KSK, which is one of the keys signing only the Key RRset, then it is necessary to do the same changes as previously on the Key RRset, that is to say to create or to delete a key and to resign the zone. Since the set of the KSK has been modified, it is necessary to realize a

synchronization with the parent zone in order to accomplish the update of DS records concerned with this change.

Our algorithm enables the installation of an automatic synchronization between the parent zone and the child zone. At any time of the rollover process, the two zones are accessible and their resource records are verifiable. Moreover the algorithm is based on mechanisms already existing of DNSsec.

### 3.1. Description of the Key Rollover Algorithm

In this part we present the principle of our algorithm, called KRO. Initially we suppose that there exists, at the time of the suppression of a KSK, at least another KSK present in the child zone and the corresponding DS resource record in the parent zone.

1. The child zone decides to change its KEY RRset by adding or deleting one or more keys. If it is a ZSK then go to step 2, else (it is a KSK) go to step 3.
2. The child zone creates/deletes a ZSK and resigns the zone. Go to step 1.
3. The child zone creates/deletes a KSK, changes its KEY RRset and resigns the zone.
4. The child zone sends a DNS notification message (NOTIFY QUERY [11]) to its parent zone to inform that its KEY RRset has been modified.
5. The parent zone receives the DNS notification message and obtains the KEY RRset of the child zone using a DNS request.
6. The parent zone checks this KEY RRset with the SIG resource records, the child zone KEY and the correspondent DS record. If the KEY RRset is valid, the parent zone sends an acknowledgement (NOTIFY RESPONSE) to the child zone and creates a Keyset.
7. The child zone asks periodically the DS record in order to check when the parent zone is resigned and confirms the creation of the DS record. If the DS record is not up-to-date, then go to step 4. If the DS record is up-to-date then go to step 1.

We should assume that there are at least two KSKs. This allows to keep the chain of trust for the checking and the validation of the resource records (step 6) at any time. Indeed if one of the KSK is removed, there remains a link between the parent zone and the child zone represented by the remaining KSK and corresponding recording DS.

### 3.2. Justifications of Choices

Our goal is to find a way to inform the parent zone that some changes have occurred in its child zone, by adding and changing as few as possible the DNSsec specifications. We choose to use Notify messages.

The Notify message is well designed for our algorithm because whether the Query/Response (Q/R) bit is set or not, this message can be understood as a notification of a key change (Query) or as an acknowledgement to the notification of a key change (Response). Moreover, even if it is possible, our method does not need to put data in this message. Thus, it is not necessary to digitally sign it.

To implement this algorithm, a modification of the behavior of the primary server of a zone about the Notify message is necessary. Indeed, the source of the Notify message during a change of key will be the primary server of the child zone (step 4). The primary server of the parent zone has to accept the message like a notification of a key change, because until DNS specifications [11] a Notify message could be exchanged only between primary and secondary name servers inside the same zone.

Another type of message could be used instead of the Notify: the Dynamic Update message [12]. The zone deciding to change its keys could send a Dynamic Update message containing these keys to the primary server of its parent zone. The transmission of data implies the protection of the message, it is therefore necessary to use SIG(0) [3] or TSIG [10] which are mechanisms that sign the complete DNS messages. This suggests that the private keys are available on the name server in order to sign this message. In the opposite case (a not signed Dynamic Update message), malicious data could be inserted in the message. Moreover, to avoid replay attacks the parent zone server must check that the keys are present in its child zone sending a DNS request. The advantage of conveying the keys in the Dynamic Update message is therefore lost.

The method using the Dynamic Update message implies the same number of messages as the method using the Notify message, with an additional load: the keys contained into the Dynamic Update message. That is why we choose to use Notify message.

The use of the Notify however implies the generation of additional messages between the primary servers of the parent and child zone: the Notify Query, the Notify Response, the demand for keys and the sending of the keys. This number of messages is limited and is not a consequent increase of the workload of a name server.

### 3.3. Security Considerations

Our method is based on existing mechanism, no new types of message must be created. From a security point of view, every message sent during the rollover phase benefits of security mechanisms included in DNSsec.

The first Notify message sent (Notify Query) in order to notify parent zone that some changes have occurred in its child zone, contains no data. There is no need to sign it.

Once this notify query message is received by the parent zone, the parent zone sends a DNSsec request to obtain the child zone KEY resource record. The response contains records and associated SIG records. Thus integrity and authentication are guaranteed. No one can change the content of the message. If a resource record is altered the associated SIG record do not match anymore.

Moreover, we do not use SIG(0) mechanism which impose to keep the private key on name server to sign messages. To store the private key on a name server bring some strong security constraints because the name server is fully accessible and the private key must be kept secret.

### 3.4. Related Work

Recently another proposition for the automation of the DNSsec key rollover was discussed in an IETF draft [9]. This document describes a mechanism for a child zone to announce to its parent zone that changes have occurred on records they share: the NS records of the child zone and the DS authenticating the child zone KSK. It uses a Notify message in which it includes the modified records and signs this message with SIG(0) RR.

This suggestion is different from KRO on these two points. This draft:

- includes RR in Notify message.
- signs the message with SIG(0).

In KRO the Key RR is retrieved from child zone to parent zone by a DNSsec request and data are authenticated with the SIG RR contained in the response. There is no need to keep private key on a name server to sign messages with SIG(0).

Recent discussion with the author of this solution brought a new idea. This kind of messages are created only when a key is changing and consequently the zone is re-signed. The idea is to create this message when

the zone is re-signed with the same off-line mechanism and the same machine.

In this point, this proposition is different with KRO because the authentication is done by the use of SIG(0) RR and the checking of this signature. In KRO the authentication is made with the checking of the signature of the KEY RRset obtained by a standard DNS request.

## 4. DISCUSSION

### 4.1. Problems with Cache Server

We can distinguish four types of entities concerned by DNSsec key rollover, when a name must be resolved: the child zone for KEY records, the parent zone for DS records, recursive cache servers and resolvers.

To be scalable, the DNS infrastructure comprises cache servers which are not authoritative on any zone. Their role is to carry out name resolutions for resolvers and to store resource records during a time defined by the Time to live field (TTL) associated with the data.

Thus a record can be duplicated into several places at the same time: on the authoritative servers and in recursive cache servers. It is necessary to be able to keep consistency between the different locations of this record during the creation of the chain of trust, since the required records for validation can be sent partly by an authoritative server and partly by a cache server. In the case of a key change and if the zone has only one KSK, the problem shown on Fig. 1 can arise.

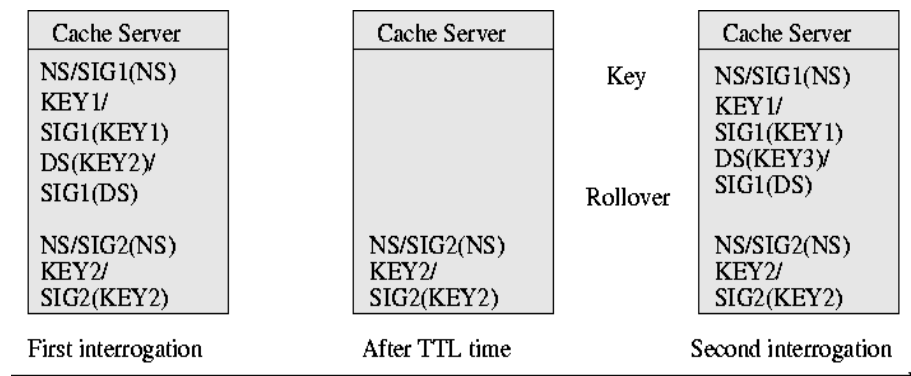


Fig 1. Fast rollover problem.

If a name is resolved just before the key rollover, all the data required for the validation of the chain of trust are present in the cache server. After a time longer than the TTL for the data of the parent zone, DS records are flushed from the cache server. If the key rollover is too quick, i.e. without checking that the old KSK is flushed from the cache server, the scenario presented on the Fig. 1 can occur after a second name request. There is a DS for KEY3 in the cache server, whereas the cache server has only KEY2 and the signature created using KEY2. If a resolver makes a name resolution using this cache server and if the resolver wants to check itself the received information, it will detect an error because the chain of trust is not verifiable anymore although all changes were correctly handled in the two zones.

To solve this problem we have modified our algorithm and added a test at step 3 and an eighth step. The added modifications are a temporal overlapping of both the new KSK and of the previous KSK.

1. The child zone decides to change its KEY RRset by adding or deleting one or more keys. If it is a ZSK then go to step 2, else (it is a KSK) go to step 3.
2. The child zone creates/deletes a ZSK and resigns the zone. Go to step 1.
3. If there are at least two KSK, the child zone creates/deletes a KSK, changes its KEY RRset and resigns the zone.

If there is only one KSK and it must be changed, the child zone creates a new KSK, changes its Key RRset and resigns its zone.

4. The Child zone send a DNS notification message (NOTIFY QUERY [11]) to its parent zone to inform that its KEY RRset has been modified.
5. The parent zone receives the DNS notification message and takes the KEY RRset of the child zone using a DNS request.
6. The parent zone checks this KEY RRset with the SIG resource record, the child zone KEY and the correspondent DS record. If the KEY RRset is valid, the parent zone sends an acknowledgement (NOTIFY RESPONSE) to the child zone and creates a Keyset.
7. The child zone asks periodically the DS record in order to check when the parent zone is resigned and confirms the creation of the DS record. If the DS record is not up-to-date, then go to step 4.
8. Since the parent zone has created DS record and has resigned its zone, the child zone is waiting  $\max(\text{parent-zone-DS-record-TTL},$

child-zone-KEY-record-TTL) time to delete (if necessary) the old KSK and the old SIG resource records created using this KSK. Go to step 1.

Before deleting the old KSK definitely, it is necessary to ensure that this KEY record and its corresponding DS record are not present anymore in a DNS cache server, otherwise the child zone could become inaccessible. If the server is waiting for this delay before deleting the KSK, we maintain the consistency of information and we keep the chain of trust intact because the cache server will have to do a new request to the authoritative server to obtain the records. Thus, it will receive the new KSK and the new DS.

## 4.2 Emergency Key Rollover

A key change can be scheduled, it is the usual case. Another key change case is when one key of the zone is compromised. This key has to be changed as quickly as possible. In such a case the change of the key is immediate in the concerned zone and is followed by the deletion of the compromised key, the deletion of signatures generated with this key and finally the zone re-signature. It is then necessary to notify the parent zone to quickly remove the DS which authenticated the compromised key.

If the zone contained only one KSK and if this key has to be changed in emergency, there is no way of waiting for  $\max(\text{parent-zone-DS-record-TTL}, \text{child-zone-KEY-record-TTL})$  time to delete the compromised key (step 8 of the algorithm). As previously, if a request for name resolution is made at this time, the data of the zone and its child zones are no longer verifiable because there is no DS in the parent zone which authenticates the new created key. Moreover, as long as the DS authenticating the compromised key remains in the parent zone file, corrupted data may be created and put into cache server using attacks like cache pollution.

## 4.3 The Bootstrap Problem

We have supposed in section 3 that the zone was already secured. If this is not the case, an authentication problem occurs: anyone wanting to create a delegation for a future unknown child zone. This authentication must be made outside of the DNSsec protocol. However if we assumed that this authentication is made by the parent zone according to a mean it chooses (PGP key, X.509 certificate, etc.) then it is possible to install the delegation in a secured and automated way (using a variant of the KRO algorithm).

## 4.4 Future Work

The KRO algorithm was initially designed for the key rollover in DNSsec. By solving the difficulties that occurred for the automation of the key rollover process and by implementing this solution, it appears that we could develop a more complete tool than KRO which would allow a total automation of the installation of a DNSsec zone and its management. From a DNS zone file it would allow to create a DNSsec zone file by generating the keys and the signatures and by notifying the parent zone of the presence of new keys. KRO could then be integrated inside this tool to manage the key rollover.

## 5. CONCLUSIONS

DNSsec is an essential service in terms of security of the DNS. DNSsec allows to secure one vulnerable and very frequently used service. However DNSsec gives also an extra workload to the administrators who want to install DNSsec. Such a workload could slow down the development of this technology.

The algorithm we present in this paper allows to completely automate the process of key rollover using existing mechanisms of DNSsec, except for a minor modification of the specifications of the Notify message. This algorithm also guarantees no loss of contact with the child zone by using a temporal overlapping of the keys during their change and by taking into account the use of cache server in DNS infrastructure.

Some of the above problems remain open, like the emergency key rollover for example. Discussions are beginning in the IETF DNSext Working Group in order to define the requirements necessary to the resolution of these problems.

The algorithm that we have just presented allows the automation of the management of a secured zone, decreasing the workload of the DNS zone administrators. The simplification of the installation of secured zones should bring a faster acceptance and deployment of DNSsec resulting into an optimal security of the Internet if this service is present in every zone.

## 6. REFERENCES

- [1] D. Atkins and R. Austein (Jun 2003), Threat Analysis Of The Domain Name System. Draft IETF, work in progress.
- [2] D. Eastlake (March 1999), Domain Name System Security Extensions. RFC 2535.
- [3] D. Eastlake (Sep 2000), DNS Request and Transaction Signatures (SIG(0)s). RFC 2931.
- [4] G. Guette and B. Cousin (March 2003), Les faiblesses du DNS. In 2ème rencontre francophone sur Sécurité et Architecture Réseaux, SAR'2003.
- [5] R. Gieben (2001), Chain of Trust. Master's Thesis, Nlnet Labs.
- [6] O. Gundmundsson (Jun 2003), Delegation Signer Resource Record. Draft IETF, last call.
- [7] P. Mockapetris (Nov 1987), Domain Names – Concept and Facilities. RFC 1034.
- [8] P. Mockapetris. Domain Names – Implementation and Specification. RFC 1035, Nov 1987.
- [9] M. StJohns (Jun 2003), Using DNSSEC-secured NOTIFY to Trigger Parent Zone Updating. Draft IETF, work in progress.
- [10] P. Vixie, O. Gundmundsson, D. Eastlake and B. Wellington (May 2000), Secret Key Transaction Authentication for DNS (TSIG). RFC 2845.
- [11] P. Vixie (Aug 1996) A Mechanism for Prompt Notify of Zone Changes (DNS NOTIFY). RFC 1996.
- [12] B. Wellington (Nov 2000), Secure Domain Name System (DNS) Dynamic Update. RFC 3007.