Algorithm for DNSSEC trusted key rollover

Gilles Guette[1], Bernard Cousin[1], and David Fort[1]

IRISA, Campus de Beaulieu, 35042 Rennes CEDEX, FRANCE
{gilles.guette, bernard.cousin, david.fort}@irisa.fr

Abstract The Domain Name System Security Extensions (DNSSEC) architecture is based on public-key cryptography. A secure DNS zone has one or more keys and signs its resource records with these keys in order to provide two security services: data integrity and authentication. These services allow to protect DNS transactions and permit the detection of attempted attacks on DNS.
The DNSSEC validation process is based on the establishment of a chain of trust between zones. This chain needs a secure entry point: a DNS zone whose at least one key is trusted. In this paper we study a critical problem associated to the key rollover in DNSSEC: the trusted keys rollover problem. We propose an algorithm that allows a resolver to update its trusted keys automatically and in a secure way without any delay or any break of the DNS service.

1 Introduction

During the last decade, the Internet Engineering Task Force (IETF) has developed the DNS security extensions (DNSSEC). The first standard document designing DNSSEC is the RFC 2535 [Eas99]. Many RFCs and drafts have updated the RFC 2535 and according to the experience given by implementations, this protocol is today enhanced [ALMR04] [AAL+04a] [AAL+04b].

The DNSSEC architecture uses public key cryptography to provide integrity and authentication of the DNS data. Each node of the DNS tree, called a *zone*, owns at least a key pair used to secure the zone records with digital signatures.

In order to validate DNSSEC records, a resolver builds a chain of trust [Gie01] by walking through the DNS tree from a secure entry point [KSL04] (typically a top level zone) to the zone queried. Each secure entry point (SEP), is statically configured in a resolver: the resolver knows at least one key of the zone which is taken as SEP, this key is called a *trusted key*. A resolver is able to build a chain of trust if it owns a secure entry point for this query and if there are only secure delegations from the secure entry point to the zone queried.

The lifespan of keys used to secure DNS zone is not infinite, because old keys become weak. Consequently, the zone keys must be renewed periodically, that is to say a new zone key is added to the zone file and an old one is deleted from the zone file. This process is called key rollover [GC03].

Key rollover only updates keys on the name server, resolvers that have configured the old key as trusted are not notified that this key has been deleted.

Consequently, the static key configuration in a resolver raises some problems of consistency between keys deployed in a zone and trusted keys configured in a resolver for this zone.

In section 2 we present the notations use in this paper and the DNSSEC validation process. Then, in section 3 we describe the trusted key rollover problem and finally we present our solution to this problem in section 4.

2 The validation process in DNSSEC

2.1 Notation

In this subsection are explained the notations used in the document.

- A DNS domain X is the entire subtree beginning at the node X.
- A DNS zone is a node of the DNS tree. A zone name is the concatenation of the node's label from its node to the root of the DNS tree. A zone contains all the not delegated DNS names ended by the zone name. For example, the zone `example.com.` contains all the not delegated names `X.example.com.` where X can be composed by several labels.
- A zone can delegate the responsability of a part of its names. For example, the zone `example.com.` can delegate all the name ended by `test.example.com.` to a new zone. This zone is named the `test.example.com.` zone (Fig 1).

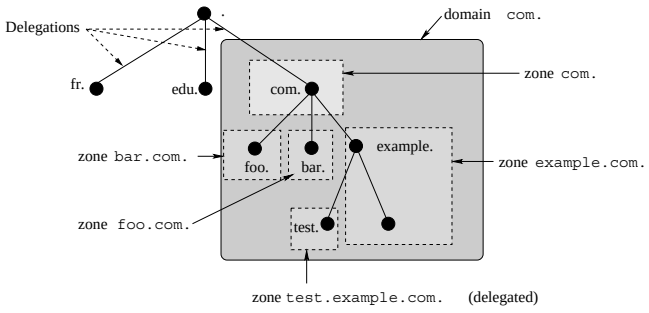

Figure1. DNS domain and DNS zone.

- RR means Resource Record, the basic data unit in the domain name system, each RR is associated to a DNS name. Every RR are stored in a zone file.
- Resource records with same name, class and type form a RRset. For example the DNSKEY RRs of a zone form a DNSKEY RRset.
- DNSKEY ($key1$) is the RR which describes the key named *key1*.
- RRSIG$(X)_y$ is the RR which is the signature of the RR X generated with the private part of key y.

– A trusted key is the public part of a zone key which has been configured in a resolver.
– A Secure Entry Point is a zone for which the resolver has configured a trusted key.

2.2 The DNS entities and definitions

Three entities with distinct role are present in the DNS architecture: the name server, the resolver and the cache server (see [Moc87,AL02]).

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

The resolver. The resolver is the local entity that receives a request from an application and sends a DNS queries to the appropriate name server. After having performed the name resolution the resolver sends the response back to the application.

The cache server. The cache server is not authoritative on any zone. The scalability of DNS is based on the use of cache servers. These cache servers only forward queries and cache the responses when these are valids.

2.3 The DNSSEC Chain of Trust

DNS security extensions define new resource records in order to store keys and signatures needed to provide integrity and authentication.

Each secured zone owns at least one zone key, the public part of this key is stored in a DNSKEY resource record. The private part of this public-key is kept secret and should be stored in a secure location. The private part of the key generates a digital signature for each resource record in the zone file. These signatures are stored in a RRSIG resource record. A resource record is considered valid when the verification of **at least one** of its associated RRSIG RR is complete. Figure 2 shows the signature verification process.

In order to verify the signature of a resource record, the resolver cyphers the RRSIG RR with the public key contained in the DNSKEY RR of the zone. If the result of this operation (`Resource Record'`) and the `Resource Record` present in the DNS response message are the same, the signature is verified. The resource record is valid.

During the signature verification process, the zone key is needed and must be verified too. In order to trust a zone key, DNSSEC uses the DNS-tree model to establish a chain of trust [Gie01] beginning from a secure entry point [KSL04] to the queried zone. To create this chain, a verifiable relation between child zone and parent zone must exist: this is the role of the Delegation Signer resource record (DS RR) [Gun03]. This record, stored in the parent zone, contains information allowing the authentication of one child zone key. Figure 3 shows the validation of a delegation.

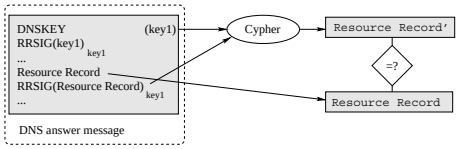

Figure2. The signature verification process.

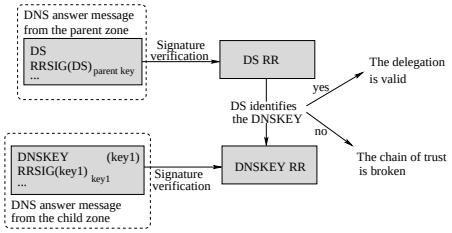

Figure3. The delegation verification process.

Once the signature of the DS RR provided by the parent zone and the signature of the DNSKEY provided by the child zone are verified, the resolver checks that information contained in one DS RR identifies one key in the child zone. If one DS RR identifies one DNSKEY RR in the child zone, one link of the chain of trust is built and the name resolution continue to secure the next link in the DNS tree. If there is no valid DS RR that identifies one valid DNSKEY RR in the child zone, the chain of trust is broken and the name resolution is unsecure.

The Delegation Signer model introduces a distinction between two types of key: the Zone Signing Key (ZSK) and the Key Signing Key (KSK). A ZSK signs all types of record in the zone file and a KSK signs only the DNSKEY RRs present in the zone file. This distinction minimizes the burden produced when a key is changed in the child zone, because only KSKs have an associated DS RR in the parent zone file. Hence, a DNSKEY RR can only be considered valid by the verification of a signature generated by a KSK.

3 The trusted key rollover problem

The validation process of resource records based on the DNSSEC chain of trust needs a secure entry point to start. The RFC 2535 [Eas99] specifies that a DNSSEC resolver must be configured with at least a public key which authenticates one zone as a secure entry point. If DNSSEC is totally deployed on the whole DNS tree, only one secure entry point is sufficient, *i.e.* the root zone. But due to deployement constraints, the current model that seems to emerge is an *island of security* model with DNS zones and DNSSEC zones at the same time in the DNS tree. An island of security is a subtree of the DNS tree totally secured with DNSSEC (each zone of this subtree has a signed zone file). Consequently, a resolver needs at least one secure entry point for the apex of each island of security in order to perform secure name resolution for any zone.

Moreover, to maintain a good level of security, zone keys have to be renewed at regular intervals, to prevent against key disclosure. And consequently, trusted keys must be updated in resolvers to keep consistency between the key in the zone file of a name server and the trusted key in the resolvers.

Currently, the rollover of a trusted key in the resolver's configuration file is done manually by the administrator, this implies risks of misconfiguration and

an interruption of the service between the moment the zone rolls its keys and the moment the administrator changes the resolver's configuration file. At the present time, there is no automated trusted key rollover. When a zone decides to renew one of its zone keys, there is no mechanism to notify the resolvers that this key is going to be removed from its zone file. When a key is removed from its zone file, all resolvers that have configured this key as trusted fails all DNSSEC validation using this key.

Without an automated procedure to notify resolvers that a trusted key is going to be removed, name resolution can fail at any time even if a chain of trust exists from the root to the resource record queried.

In the next section, we propose a modification of the DNSKEY resource record and an algorithm to automatically update the trusted keys present in the resolver configuration file.

4 A mechanism for trusted key rollover

4.1 The DNSKEY resource record wire format

Figure 4 shows the DNSKEY resource record format.

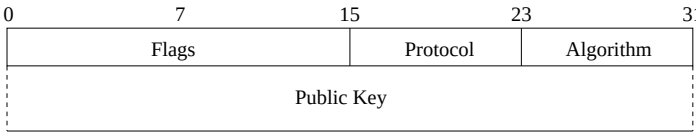

Figure4. The DNSKEY resource record wire format.

Only two bits of the Flags field are used. These bits are the seventh bit and the fifteenth bit. Bit 7 of the Flags field is the Zone Key flag. If bit 7 is set, then the DNSKEY record holds a DNS zone key. If bit 7 has value 0, then the DNSKEY record holds some other type of DNS public key (such as a public key used by TKEY [Eas00]). Bit 15 of the Flags field is the Secure Entry Point flag, described in [KSL04]. If bit 15 has value 1, then the DNSKEY record holds a key intended for use as a secure entry point. The other bits of the field, bits 0-6 and 8-14 are reserved: these bits must have value 0.

The Protocol field specifies the algorithm allowed to use the DNSKEY RR. Since the publication of the RFC 3445 [MR02] this field must have value 3.

The Algorithm field identifies the public key's cryptographic algorithm and determines the format of the Public Key field that holds the public key material. The format depends on the algorithm of the key being stored.

We propose to call the *under changes* bit, the first of the reserved bits of the Flag field and to give this bit the following meaning: 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.

4.2 The automated trusted key rollover algorithm

When a zone renewed one of its keys, it sets the *under changes* bit in the DNSKEY RR containing this key. Piece of advice for the duration of the rollover period must be found in [KG04].

During the rollover period, when a resolver asks for the DNSKEY RR, it retrieves the resource record with its *under changes* bit set. Three distincts periods could be defined for each key: before the rollover of the key, during the rollover of the key and after the rollover of the key. Before and after the rollover, the key is not under changes so the bit is not set. During the rollover the *under changes* bit is set to one.

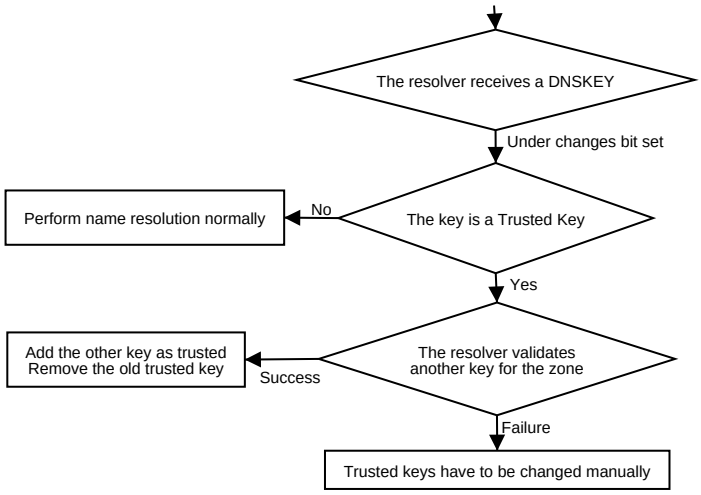

Figure5. The *best case* resolver behavior.

When a resolver receives a DNSKEY RR with the *under changes* bit set, it 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 (see section 4.3). 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 can perform no name resolution on a zone during the rollover period of a trusted key. This implies that the trusted key configured in such resolver may have been renewed and removed from the zone file during this period of *resolver inactivity*.

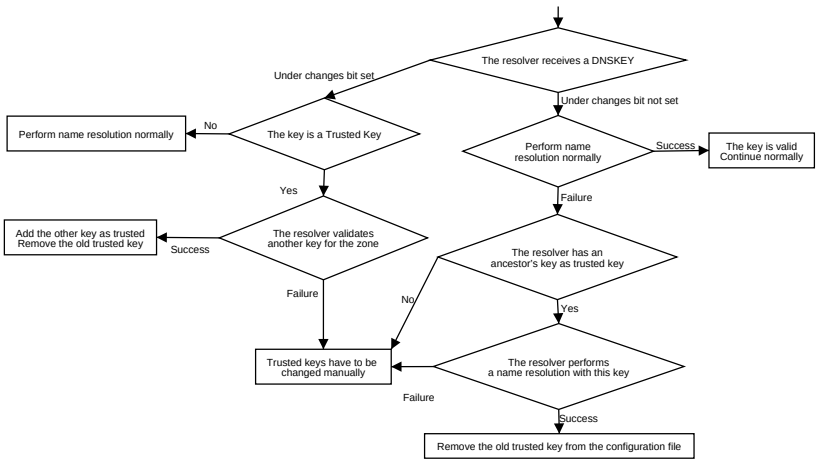

Figure6. The complete resolver behavior.

Figure 6 shows the two cases that can arise: 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 not such a trusted key. The left part of the Figure 6 is the *best case* resolver behavior presented in Figure 5, the right part takes into account the reception of a DNSKEY RR with the *under changes* bit not set.

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 performed a DNSSEC resolution either because it has no valid trusted key or because the resolution is not secure (some data are retrieved from unsecured DNS zone) the resolver can not update its configuration file and the trusted key update must be made manually.

Algorithm presented on Figure 6 shows that in some cases a manual administrative action is needed and a totally automated solution does not exist. This is due to possible misconfiguration in any DNS zone or due to the possibility for a resolver to not have sent query to a zone during its rollover period, and hence, to not have received the notification of key changes.

4.3 Security considerations and validation of a new trusted key

Automated update of trusted keys in a resolver is a critical mechanism for DNSSEC and must be resistant to compromised key. An attacker that owns a compromised key can try to place new fake keys as trusted in a target resolver using the automated trusted key rollover mechanism, as describe now.

Firstly, the target resolver has a trusted key set including the compromised key. Then, the attacker sends a forged message containing the compromised key having the *under changes* bit set, new fake keys and their signatures. Finally,

the resolver checks the signatures of the keys, one of these keys is trusted (but compromised) and validates some signatures. The resolver discards the old key and accepts the new fake keys sended by the attacker.

To protect the automated trusted key update from this attack, we define a requirement for the acceptance of a new key depending on the security level that the resolver's administrator wants.

To accept a new key, the resolver checks the signatures of the DNSKEY RRset it has received. The normal validation process succeeds if **at least one signature** is valid, but this is not enough to ensure security because the compromission of only one key allows the attack on the resolver to be fruitful. So to be resistant to key compromission the resolver must accept a new key only if k signatures generated by the trusted key owns by the resolver for this zone are valid. If one of this signature is not correct the resolver rejects the new key.

If the resolver has n trusted keys for a given zone, our automated trusted key rollover resists to $n - k$ compromised keys. Because, an attacker that owns $n - k$ compromised key for this zone can only generate $n - k$ valid signatures and the resolver will accept new keys only if there are n valid signatures.

Some local policy parameter may be defined by administrators to ensure a sufficient level of security. This parameter defines the number of trusted keys an administrator wants to configure to a given zone. If an administrator wants a low security level, he sets this parameter to one and configures only one trusted key for a zone. But, if he wants a higher security level he sets this parameter to any number included between 2 and the number of keys present in this zone.

Moreover, a resolver removes or accepts keys as soon as the DNSKEY RRset is received, our method does not implies extra delays or time constraints for resolvers or servers during the rollover period.

5 Related work

5.1 Description

Recently, Mike St Johns has published an IETF draft [StJ04] describing a method for automated trusted key rollover. This draft 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 resolver's trusted key set. This revocation is immediate. Figure 7 shows the principles of this method.

Firstly, the resolver sends a query about a DNS name ① and receives DNSKEY RR with the revocation bit set. The revocation is immediate and the resolver updates its trusted keys set ②. To avoid attack when a key is compromised, the resolver starts a timer of 30 days and keeps a trace of all the keys that signed the DNSKEY RRset ③. If the resolver received 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 keys and resets the timer.

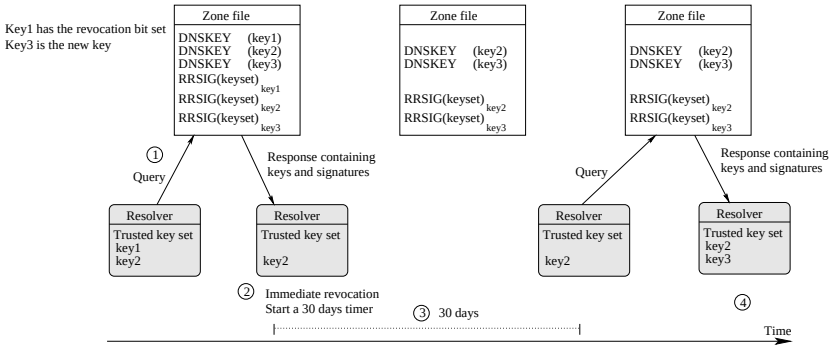

Figure7. Acceptation method with 30 days timer.

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

5.2 Comparison

The major problem with the method described in the draft [StJ04] is that it introduces extra delay (30 days) and requires a lot of management precaution 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.

Moreover, with this method, when a server creates a new key, this key is pending and can not be used during 30 days after its creation. This implies that if a zone owns n keys and rolls a keys every $\frac{30}{n-1}$ days, all resolvers will be unable to update its trusted key set for this zone.

Our method does not suffer from this problem because acceptance and revocation of keys are made as soon as the DNSKEY RRset is received. Our method does not imply changes or constraints for the DNS authoritative server management.

6 Conclusion

In this paper we have presented the trusted key rollover problem and we have proposed a protocol modification and an algorithm for resolvers to solve the trusted key rollover problem. The automation of the trusted key rollover allows to avoid break in the DNS service due to misconfigurations and allows to minimize the work overload for administrators. We have exposed that in some case the trusted key rollover cannot be automated and the modification of the configuration file must be done manually. These cases arise seldom and are limited

to machines that do not perform name resolution during the rollover period. Moreover, the algorithm described in this paper is resistant to compromised key and places the choice of security level in the resolver's administrator side, which is a natural way to protect a resolver.

References

[AAL+04a] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Protocol Modifications for the DNS Security Extensions. Draft IETF, work in progress, Sep 2004.

[AAL+04b] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose. Resource Records for the DNS Security Extensions. Draft IETF, work in progress, Sep 2004.

[AL02] P. Albitz and C. Liu. *DNS and BIND.* O'Reilly & Associates, Inc., Sebastopol, CA., 4me edition, Jan 2002.

[ALMR04] R. Arends, M. Larson, D. Massey, and S. Rose. DNS Security Introduction and Requirements. Draft IETF, work in progress, Sep 2004.

[Eas99] D. Eastlake. Domain Name System Security Extensions. RFC 2535, Mar 1999.

[Eas00] D. Eastlake. Secret Key Establishment for DNS (TKEY RR). RFC 2930, Sep 2000.

[GC03] G. Guette and O. Courtay. KRO: A Key RollOver Algorithm for DNSSEC. In *International Conference on Information and Communication (ICICT'03)*, Nov 2003.

[Gie01] R. Gieben. Chain of Trust. Master's Thesis, NLnet Labs, 2001.

[Gun03] O. Gundmundsson. Delegation Signer Resource Record. RFC 3658, Dec 2003.

[KG04] O. Kolkman and R. Gieben. DNSSEC operational practices. Draft IETF, work in progress, Apr 2004.

[KSL04] O. Kolkman, J. Schlyter, and E. Lewis. Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag. RFC 3757, Apr 2004.

[Moc87] P. Mockapetris. Domain Names - Concept and Facilities. RFC 1034, Nov 1987.

[MR02] D. Massey and S. Rose. Limiting the Scope of the KEY Resource Record (RR). RFC 3445, Dec 2002.

[StJ04] M. StJohns. Automated Updates of DNSSEC Trust Anchors. Draft IETF, work in progress, Mar 2004.