GDS Resource Record: Generalization of the Delegation Signer Model

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. Domain Name System Security Extensions (DNSSEC) architecture is based on public-key cryptography. A secure DNS zone has one or more keys to sign its resource records in order to provide two security services: data integrity and authentication. These services allow to protect DNS transactions and permit the detection of attacks on DNS.
The DNSSEC validation process is based on the establishment of a chain of trust between secure zones. To build this chain, a resolver needs a secure entry point: a key of a DNS zone configured in the resolver as trusted. Then, the resolver must find a path from one of its secure entry point toward the DNS name to be validated. But, due to the incremental deployment of DNSSEC, some zones will remain unsecure in the DNS tree. Consequently, numerous trusted keys should be configured in resolvers to be able to build the appropriate chains of trust.
In this paper, we present a model that reduces the number of trusted keys in resolvers and ensures larger secure access to the domain name space. This model has been implemented in BIND.

1 Introduction

Domain Name System (DNS) [1,2,3] is a distributed database mostly used to translate computer names into IP addresses. The DNS protocol does not include any security services such as integrity and authentication, which let the protocol vulnerable to several attacks [4,5,6]. Therefore the Internet Engineering Task Force (IETF) has developed the DNS security extensions (DNSSEC).

DNSSEC architecture [7,8,9,10] 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 information with digital signature.

In order to validate DNS data with DNSSEC, a resolver builds a chain of trust [11] by walking through the DNS tree from a secure entry point [12] (*i.e.*, a *trusted key* statically configured in the resolver, typically a top level zone) to the zone queried. 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.

Because of the long[1] and incremental deployement of DNSSEC, resolvers will have to keep many trusted keys. The number of trusted keys needed for a

[1] Transition duration could be long: for instance IPv6 deployement lasts since 1995

single resolver may be huge and not easy to store. Moreover, these zone keys are periodically renewed and the resolver must update its trusted keys set to keep consistent the keys.

In this paper we propose the creation of a new resource record allowing to reduce the number of trusted keys needed in a resolver. In Section 2, we present the definitions used in this paper and an overview of the DNSSEC processing. In Section 3, we expose the problem of the islands of security model for the resolvers. Section 4 defines the General Delegation Signer Resource Record and its management. Then, in Section 5 we discuss the pros and cons of our method.

2 Validation process in DNSSEC

2.1 Definitions

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

(1) A DNS domain X is the entire subtree included into the DNS tree beginning at the node X. (2) A DNS zone is a node of the DNS tree. A zone name is the concatenation of the node's labels 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. (3) A zone can delegate the responsability of a part of its names. The zone `example.com.` can delegate all the names ended by `test.example.com.` to a new zone. This zone is named the `test.example.com.` zone. (4) RR means Resource Record, the basic data unit in the domain name system. Each RR is associated to a DNS name. Every RR is stored in a zone file and belongs to a zone. (5) Resource records with same name, class and type form a RRset. For example the DNSKEY RRS of a zone form a DNSKEY RRset. (6) DNSKEY ($key1$) is the RR which describes the key named *key1*. (7) RRSIG$(X)_y$ is the RR which is the signature of the RR X generated with the private part of key y. (8) A trusted key is the public part of a zone key, configured in a resolver. (9) A Secure Entry Point is a zone for which the resolver trusts a key.

2.2 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 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 1 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 of the zone contained in the DNSKEY RR present in the DNS response message. If the result of this operation, called `Resource Record'` in figure 1 is equal to `Resource Record` present in the DNS response message, the signature is verified. Thus, the resource record is valid.

During the signature verification process, the zone key is needed and must be verified too. This allows to avoid the use of a fake key sent in a message forged by a malicious person. To trust a zone key, DNSSEC uses the DNS-tree model to establish a chain of trust [11] beginning from a secure entry point [12] 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) [13]. This record, stored in the parent zone, contains information allowing the authentication of one child zone key. Figure 2 shows the validation of a delegation.

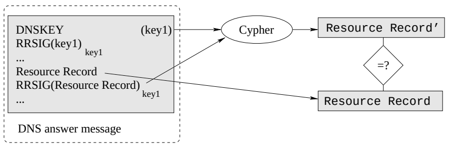

Fig. 1. The signature verification process.

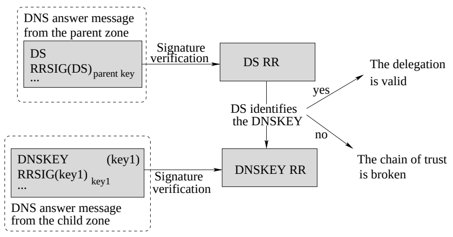

Fig. 2. The delegation verification process.

Once the signatures (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 should progress 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.

3 DNSSEC Deployement and Islands of Security Model

Using DNSSEC is not mandatory for zone administrators, they are free to deploy DNSSEC or not. Indeed, some administrators evaluate the overhead to deploy DNSSEC too important compared to the need to secure their zone with DNSSEC. Moreover, deploying DNSSEC software can raise some compatibility problems. Consequently, some parts of the DNS tree may be secure with DNSSEC while some other parts may be unsecure (DNS only). This leads to an *islands of security* model. An island of security is a subtree of the DNS tree entirely secured with DNSSEC. The incremental deployment of DNSSEC implies some remaining unsecure zones in the DNS tree and hence unsecure delegations. As a resolver may send queries about any DNS name, it should be able to perform secure name resolution about any zone in any existing islands of security. The first naive solution is to configure in each resolver the apex key of all existing islands of security as trusted keys. Two major reasons prove this naive solution can not be implemented: the set of keys needed is hard to manage (the DNS tree is composed by several million of zones). And the zone keys are periodically changed in order to resist to cryptoanalysis attack and should be updated in all resolvers.

The idea discussed in this paper reduces the number of trusted keys needed in a resolver. We define a new resource record: the General Delegation Signer Resource Record (GDS RR) that is a generalization of the Delegation Signer Resource Record (DS RR) [13]. The DS RR makes the link between parent and child zones. With the GDS RR, we generalize this to a link between islands of security. This generalization allows to avoid the *gap of security* created by an unsecure zone and reduce the number of trusted keys needed for a resolver.

4 GDS resource record

The GDS resource record is a generalization of the the DS RR and for compatibility reasons we decide to create a new RR and to copy the DS RR format. This allows to keep compatibility with the current validation process implemented in DNS software. Old resolvers that understand only DS RR can continue to validate RRs, even when GDS RRs are present. The management of the GDS RR is slightly different from the DS management. The GDS resource record contains the same four fields than these contained in a DS RR: key tag, algorithm, digest type and the digest of a public key. The key tag is an identifier of the public key. The algorithm field is the algorithm of the key. The digest type identifies the digest algorithm used. The digest field contains the digest of the DNSKEY RR. This format allows concise representation of the keys that the secure descendant will use, thus keeping down the size of the answer for the delegation, reducing the probability of DNS message overflow.

4.1 Management

In the DS management the only involved entities are one zone and its parent zone. Figure 3 shows the current steps to create a DS RR when it is possible. When a zone creates new keys (the zone becomes secured or changes a subset of its keys), this zone notifies its parent zone in order to create DS RRs for these keys. If the parent zone is secure, parent and child zones exchange data needed for the DS RR creation(signed DNSKEY RR). The parent zone checks the received material and then stores the new DS RR in its zone file. A secure link is created between parent and child zones. The delegation signer model is limited to a direct link between a parent zone and one of its child zone.

In the previous algorithm, the requirement for a zone to have a DS RR referencing one of its keys is to have a secure parent. When the parent zone is unsecure it does not store any DS RR for its secure child zones. Consequently, its secure child zones are apex of islands of security. The keys of these zones must be configured as trusted by resolvers that want to perform secure name resolution. The GDS RR solves this problem of additional trusted keys configuration. If a secure zone does not have a secure parent but has a closest secure ancestor (CSA), a GDS RR stored in the CSA creates a secure link between the CSA and the secure zone. GDS RRs stored in the CSA zone file allow to authenticate keys of the secure zone and hence no additional trusted keys configuration is needed in resolvers. The authentication of keys is the same as presented in figure 2 for DS RR. A resolver sends a query about GDS RR identifying the key. It verifies the signatures of the DNSKEY RR and of the GDS RR. Finally, it verifies that the GDS RR identifies the key contained in the DNSKEY RR.

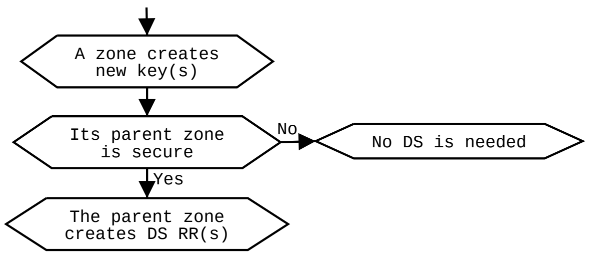
A zone creates new key(s)
Its parent zone is secure
No
No DS is needed
Yes
The parent zone creates DS RR(s)

Fig. 3. The current steps of DS creation.

A modification of the DS creation algorithm is needed for the creation of the GDS RR when a secure zone creates keys. Two operations on keys can imply the creation of GDS RR: a key rollover in a zone or a unsecure zone becomes secure. Figure 4 shows the steps to follow during the creation of GDS RRs.

The first steps of this algorithm are similar to the current algorithm, because when a DS RR authenticates a key of a given zone, no GDS RR is needed for this key. If the zone that have created new key does not have a secure parent, this secure zone must search its closest secure ancestor. Once, the zone finds its CSA (queries about DNSKEY RR on ancestor zones are sufficient to decide if an ancestor zone is secure or not), the zone exchanges its public keys with its CSA to create GDS RR(s) in the CSA zone file. For the creation of GDS RR in case of a zone key rollover, the previous steps are sufficient (see subsection 4.2).

In case of the creation of a new secure zone, to keep the unicity of secure chain, the CSA must transmit to the new secure zone all the GDS RRs of the zones which are in the subdomain of the new secure zone. When the new secure zone receives these informations from its CSA, the new secure zone can create the secure delegation for zones in its subdomain.

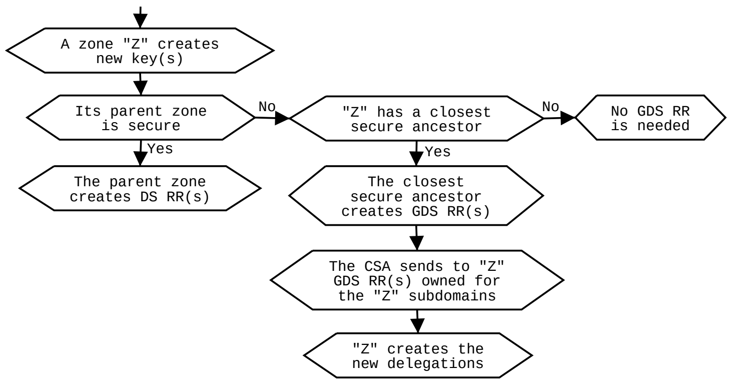
A zone "Z" creates new key(s)
Its parent zone is secure
No
Yes
The parent zone creates DS RR(s)
"Z" has a closest secure ancestor
No
No GDS RR is needed
Yes
The closest secure ancestor creates GDS RR(s)
The CSA sends to "Z" GDS RR(s) owned for the "Z" subdomains
"Z" creates the new delegations

Fig. 4. The steps of GDS creation.

The new secure zone has to send queries to its child zones only to verify that RRs are effectively present. No other transactions are needed, because all data needed for the creation of secure delegation are sent by its CSA.

4.2 Proof of concept

During a key rollover. The current key management in DNSSEC implies that when a zone changes one of its key signing key (KSK), the DS RR identifying this KSK must be updated to keep the chain of trust safe [14]. This management is the same for the GDS RR. When a zone having a CSA, changes one of its KSK the GDS RR identifying this key must be updated. So, the secure zone must contact its CSA to update the GDS RR. The only change to make in the CSA zone is the update of the GDS RR. The CSA does not have to transmit GDS RR, because nothing is changed in the DNS-tree topology.
A zone becomes secure. When an unsecure zone becomes secure GDS RRs must be created, the CSA of the zone should manage this set of GDS RRs. Figure 5 shows the transmission of the GDS RRs between the CSA and the new secure zone. Firstly, the new secure zone finds its CSA, and provides its keys. Then, the CSA creates GDS RRs for the keys of the new secure zone. Finally, the CSA transmits the GDS RR it owns for the descendants of the new secure zone, because the new secure zone becomes the CSA of these descendant. The new secure zone examines the GDS RRs received. If a GDS RR identifies a key of one of its direct child, the new secure zone creates a DS RR and deletes the GDS RR. The other GDS are kept in the zone file of the new secure zone.

We can notice that the chain of trust is keeping safe, a secure path always exists between the islands of security. A resolver can perform secure name resolution about all islands of security present in the tree with only one key of the root zone configured as trusted. In the current model, a resolver that wants to perform secure name resolution about all the islands of security present in the

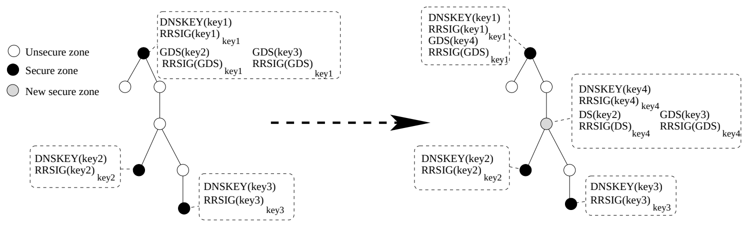

Fig. 5. The GDS RRs update and transmission.

tree must have configured one trusted key for every islands (for instance in figure 5 $key1$, $key3$ and $key4$ as trusted.

5 Pros and cons

The proposition described in this paper solves the scalability problem caused by the large number of trusted keys needed in a resolver in order to perform secure name resolutions. With GDS, only the keys of the apex of the upper islands of security have to be configured in the resolver. With this proposition the number of trusted keys in a resolver is easily manageable. Consequently, the incremental deployment of DNSSEC is not anymore a problem to its early use. GDS has been implemented during the IDsA project [15]. This implementation consists on a modification of BIND software [16]. The modified resolver implemented during the IDsA project is able to validate resource records trusting only the root key and following the secure delegations. This resolver validates the secure link existing between islands of security by verifying the GDS RRs.

Moreover, as GDS does not change anything in the existing DNSSEC specification (it does not change the management of existing RRs), it does not raise any compatibility problem. Old DNS softwares just ignore resource records they do not understand. Hence, name resolutions performed with old DNS software succeed even if GDS RRs are present in some zones. The GDS RR provides additional security between the islands of security because it emulates an entire secure tree among secure zones. For instance, experiments show that, when GDS is used, all DNS queries on a secure zone can be validated if one judicious secure entry point has been configured in every resolver despite the presence of numerous unsecure zones.

The main drawback of GDS is the additionnal records stored in the zone files.Possible conflict may occur with the zone local policy if the zone does not want to take care about (except from its child zones).

6 Conclusion

DNSSEC provides integrity and authentication to the Domain Name System. In this paper, we have described a model solving the problem of the large number of trusted keys needed in DNSSEC resolvers to perform secure name resolution. GDS RRs provide secure links between islands of security and reduce the number of trusted keys needed in a resolver. By storing GDS RRs in zone file, the trusted keys management becomes easier for the DNS administrator because of the very small number of keys needed.

Moreover, the GDS model provides a gain of security. Even without a trusted key configured for a given secure zone, a resolver can perform secure name resolution about this zone. The chain of trust is provided automatically by the GDS RRs linking the islands of security.

The GDS RR is implemented and does not raise any compatibility problem with standard DNSSEC. Its use implies a very low overhead, an easier management of the resolver trusted keys set and ensures a larger secure access to the domain name space.

References

1. Mockapetris, P.: Domain Names - Concept and Facilities. RFC 1034 (1987)
2. Mockapetris, P.: Domain Names - Implementation and Specification. RFC 1035 (1987)
3. Albitz, P., Liu, C.: DNS and BIND. fourth edn. O'Reilly & Associates, Inc., Sebastopol, CA. (2002)
4. Bellovin, S.M.: Using the Domain Name System for System Break-Ins. In: Proceedings of the fifth Usenix UNIX Security Symposium, Salt Lake City, UT (1995) 199–208
5. Schuba, C.L.: Addressing Weaknesses in the Domain Name System. Master's Thesis, Purdue University, Department of Computer Sciences (1993)
6. Atkins, D., Austein, R.: Threat Analysis Of The Domain Name System. RFC 3833 (2004)
7. Eastlake, D.: Domain Name System Security Extensions. RFC 2535 (1999)
8. Arends, R., Larson, M., Massey, D., Rose, S.: DNS Security Introduction and Requirements. Draft IETF, work in progress (2004)
9. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: Protocol Modifications for the DNS Security Extensions. Draft IETF, work in progress (2004)
10. Arends, R., Austein, R., Larson, M., Massey, D., Rose, S.: Resource Records for the DNS Security Extensions. Draft IETF, work in progress (2004)
11. Gieben, R.: Chain of Trust. Master's Thesis, NLnet Labs (2001)
12. Kolkman, O., Schlyter, J., Lewis, E.: Domain Name System KEY (DNSKEY) Resource Record (RR) Secure Entry Point (SEP) Flag. RFC 3757 (2004)
13. Gundmundsson, O.: Delegation Signer Resource Record. RFC 3658 (2003)
14. Guette, G., Courtay, O.: KRO: A Key RollOver Algorithm for DNSSEC. In: International Conference on Information and Communication (ICICT'03). (2003)
15. IDsA: Infrastructure DNSSEC et ses Applications. http://www.idsa.prd.fr (2004)
16. ISC: Berkeley Internet Naming Daemon. http://www.isc.org (2004)