# Key revocation system for DNSSEC

Gilles Guette
IRISA
Campus Universitaire de Beaulieu
35042 Rennes Cedex, France
Email: gilles.guette@irisa.fr

***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. In this paper, we show how DNSSEC can be attacked using compromised keys and the consequences of such attacks. Then, we propose a new revocation scheme for DNSSEC based on two new resource records. There is currently no revocation system defined in the DNSSEC standard.**

## I. 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 has developed the DNS security extensions (DNSSEC).

DNSSEC [8], [13], [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 indicating the kind of data it contains. For example, a DNSKEY RR contains a zone key, an 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 [12] 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.

This paper shows that compromised key attack is a critical threat and presents a key revocation system for DNSSEC. In Section II, we describe DNSSEC principles, the secure name resolution process and we present briefly consequences of a compromised key in DNSSEC. Then, in Section III, we present our key revocation service for DNSSEC. Note that there is currently no revocation system provided by DNSSEC. The point of view of the standard is that key management is local to a zone. If a key is compromised, the administrator of the concerned zone is responsible and must find a solution. This solution is not provided by the DNSSEC protocol. Then we study the security of our revocation service in Section IV.

## II. DNSSEC architecture and secure name resolution

The DNS is a distributed tree-based database. The DNS tree is divided into domains and zones.

### A. Domains and zones

A domain is a subtree of the DNS tree. The name of a domain is the concatenation of all node's label from the root of the subtree to the root of the DNS tree. Since two nodes having the same parent, 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.

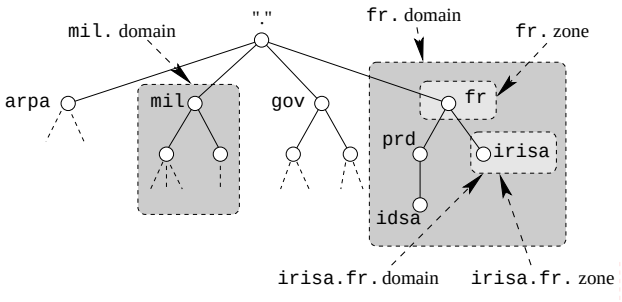



Fig. 1. DNS domains and DNS zones.

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.

### B. DNS entities

Three entities with distinct roles are present in the DNS architecture: the authoritative name server, the cache server and the resolver (see [1]).

*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 resource records contained in its zone file.

*2) The cache server:* A cache server is not authoritative on any 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 it previously received or forwards the query to the most appropriated authoritative server it knows. Using cache servers minimizes the burden on authoritative servers [14], [7].

*3) The resolver:* The resolver is the local entity (the DNS client) 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.

### C. DNSSEC chain of trust

DNS security extensions use public key cryptography and define 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 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 an 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.

*1) Two types of DNSSEC keys:* There are two ways for a resolver to trust a zone key. Either this key is configured in the resolver, this is a *trusted key* (also called a *secure entry point*), or the resolver trusts a Delegation Signer (DS) resource record [13] 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. As a DNSKEY RR is self-signed, it is the DS RR signed by the parent zone keys that authenticates the child DNSKEY RR.

A DS resource record creates also a dependency between parent and child zones. Indeed, when a key is renewed in the child zone, the DS RR that authenticates this key 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 Delegation Signer (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 resource records in the zone file.

Even though DS identifies two roles for keys, KSK and ZSK, there is no requirement that zones 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 [13].

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

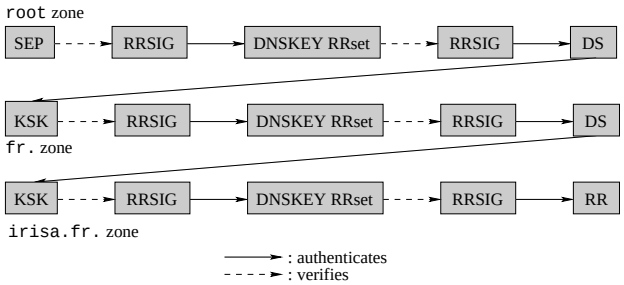



Fig. 2. Establishment of a chain of trust.

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 keys (the DNSKEY RRset) and ZSKs allow to trust other resource records of the zone.

### D. The compromised key problem

Zone keys are used in every step of a name resolution, we have shown in [11] that one compromised zone key threats all the zones under the compromised zone, even if these zones are safe.

Indeed, an attacker that owns a compromised key for a given zone is able to forge any resource record for this zone and to generate signatures for these RRs. And then, send these RR with their associated signature to the target cache server or to a resolver is enough to provide false information. Moreover, with this key the attacker can generate an entire false zone and its delegations. It can also create false or forged child zone and so on as it owns a secure entry point in the DNS tree. Hence, when an attacker owns a compromised zone key,

he is able to create a false subtree having this zone as root and then send these false information to its targets. Without a revocation system, targets cannot easily detect the attack because all information given to these targets is cryptographically correct.

We have also given in [11] formulas allowing to calculate the efficiency period of a compromised key attack. The result of this study shows us the usefulness of a revocation service in DNSSEC, because there is no such service designed in this protocol.

The DNS is designed in a distributed way and each zone is under the responsibility of an administrator. The point of view of the standard is that when a key is compromised, the administrator of the zone must find a solution and this solution should not provide (if possible) additional workload to other zones administrator. Keeping a revocation solution related to the compromised zone is maybe one of the major reasons why there is no revocation service in DNSSEC. Nevertheless, finding solutions related to the compromised zone is not sufficient because of cache servers that can have compromised resource records stored in memory. Flush the zone file from compromised records do not flush the cache servers memory.

### *E. The cache server problem from a security point of view*

An important fact to take into account is the presence of cache server in the DNS architecture. The cache servers are the reason why the DNS scales so well. In the current DNS design a cache server just forwards queries it has previously received and keep in memory the RR contained in the answers. Then, if a cache server receives a query about records it has in memory, it answers with this records without forwarding the query to an authoritative name server again. Hence, a cache server can replay old records it has in memory without verify the freshness of these records. And if the zone file has been updated and some records removed from the zone file, it may exist cache servers able to send resource records that no longer exist in the zone files.

With DNSSEC the same behavior appears. A cache server can reply to a DNSSEC query with old records (DNSKEY RR for instance), without checking that this records are always present in the zone. This could seem of less importance when it is just a host name or an IP address change (but it is at least an availability problem), nevertheless from a security point of view this is a major problem in case of key compromission. If a cache server delivers compromised DNSKEY RR even if the zone file has been updated, attacks are possible.

From the point of view of the standard describing DNSSEC, when a compromised key is detected, the administrator of the compromised zone is responsible of removing this key from its zone file, no additional mechanism is present in DNSSEC to revoke keys. But, with cache servers in the architecture, removing keys from the zone file is not sufficient because cache server can send old and compromised DNSKEY RR to client resolver.

The different approaches that can be designed are:

- the name servers notify cache servers when a key is revoked. This means all name servers knows all the cache servers. This is clearly impossible to deploy.
- The cache servers ask periodically name servers to know if a key has been revoked. This lets a vulnerability windows during which the cache servers can use a compromised key.
- The cache servers cannot cache dedicated revocation records. Hence, the cache server are forced to check if a received DNSKEY is revoked or not. This can pose some problem of name server overload.

During the reviewing process of this paper, Osterweil et *al*. have published a revocation system for DNSSEC [15]. They define a new resource record called REVKEY that only stores one compromised key. This record is self signed. Their solution is based on a particular use of the serial number included in the SOA RR. To notify resolvers from a zone state change they also define lightweight control message containing a timestamp and a signature. The signature is generated by the compromised key. But, the management of this type of messages are not clearly discussed, in particular how the timestamp is managed by the cache server and what happened with the signature if there is no compromised key. Even if the two solutions have similarities, it is difficult to compare with each other.

## III. DNSSEC KEY REVOCATION SYSTEM

In this section, we describe a novel revocation service for DNSSEC. This revocation service is supporting by two new resource records. We show how these resource records are used during the name resolution process. We took as a starting point the Delegation Signer (DS) Model to design our revocation system. Each zone owns a Key Revocation List Resource Record (KRL RR) containing the compromised keys and its parent zone owns a List Authentication Resource Record (LA RR). The LA RR authenticates one KRL RR in the same way as a DS RR in the parent zone authenticates a DNSKEY RR in the child zone.

We decide to create two records for different reasons, first to follow the delegation signer model and the spirit of DNSSEC. Then, the revocation list is under the responsibility of a zone not of its parent zone, this argues to store the KRL RR in the zone itself.

Compromised keys are stored in a KRL RR. This record must be used by DNSSEC clients to check compromised keys. As other resource records, a KRL RR is signed with the zone keys. These signatures should provide integrity and authentication for the KRL RR.

Nevertheless, a malicious person with a compromised key can generate forged resource records for the compromised zone. It can forge a false revocation list, sign it with the compromised key and pollute a cache server.

Therefore, we can conclude that signing the KRL RR with the zone key is not sufficient to prevent from

compromised key attacks. That is why we have designed the LA RR. Indeed, the LA RR contains the digest of the revocation list, is stored in the parent zone and signed with the parent zone key. If an attacker owns a compromised key and creates a fake revocation list, then the digest obtained with the fake list is different from the digest contained in the associated LA RR. As the attacker does not have a compromised key for the parent zone, he cannot create the associated fake LA RR.

This design with two records reduces the space storage needed in the parent zone. Indeed, we could directly store the KRL RR in the parent zone and sign it with the parent zone key. But, as the parent zone has many child zones, the space needed will become prohibitive. With a LA RR, the parent zone stores only one RR per child zone, and this RR has a constant size as a DS RR.

Moreover, if a signature of a revocation RR expires but the revocation list is not modified, each zone can re-sign their records (LA or KRL) without any data transmission (since there is no need to update the LA RR). This would not be the case if the KRL was signed by the parent zone and stored in the child zone, for example.

### A. The Key Revocation List Resource Record

The KRL RR contains a list of compromised keys of a given zone and all the needed information to use this list. Figure 3 details the KRL RR format.

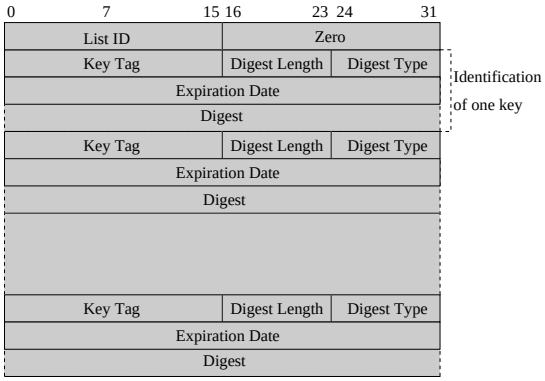



Fig. 3. The Key Revocation List Resource Record (KRL RR).

The revocation list is identified by a serial number kept in the *List ID* field. When the list is modified, the serial number is incremented. The following two bytes are set to zero and kept for future use. The revocation list must contain a set of information identifying each key. This functionality is already provided in DNSSEC by a DS RR identifying one DNSKEY RR. Hence, we have used the same format to identify the list. We have slightly modified the data structure replacing the *algorithm* field by a *Digest Length* field. This allows to easily go to one key identifier to another without checking the type of digest algorithm used to find the digest length.

The expiration date field contains the date until which the key can be removed from the revocation list. That is to say (see [11] for more details), if the compromised key is a KSK:

$$\max_{k\in K, p\in P}(\mathrm{expiration}(\mathrm{RRSIG}(\mathrm{DNSKEY\ RRset})_{\mathrm{k}}) \\ \mathrm{expiration}(\mathrm{RRSIG}(\mathrm{DS_c})_{\mathrm{p}})) \quad (1)$$

where $K$ is the set of non-compromised KSKs, $P$ is the set of ZSK of the parent zone and $DS_c$ is the DS RR authenticating the compromised key. The notation *expiration* denotes the value of the expiration time field in the RRSIG RR.

If the compromised key is a ZSK:

$$\max_{k\in K}(\mathrm{expiration}(\mathrm{RRSIG}(\mathrm{DNSKEY\ RRset})_{\mathrm{k}})) \quad (2)$$

This expiration date is calculated when a key is added to the revocation list and limits the size of the list. Indeed, there is no need to keep all compromised keys indefinitely. With DNSSEC, as soon as the chain of trust is broken, a compromised key becomes useless.

The *Digest Field* contains a hash of the key obtained by the digest algorithm specified in the *Digest Type* field. If a resolver or a cache server find a match between the key tag of a key in the revocation list and a key tag of a key it uses, then it verifies that the hash is the same. In this case, the key is a revoked one. The DNS client must react immediately (see section III-C).

### B. The List Authentication Resource Record

The LA resource record contains data authenticating a revocation list. This record is stored in the parent zone. Figure 4 shows the format of a LA RR.

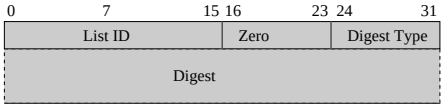



Fig. 4. The List Authentication Resource Record (LA RR).

The *List ID* field is the identifier of the list authenticated by the LA RR. It links a LA RR with a KRL RR. The next byte is set to zero and reserved for future use. The *Digest Type* field contains the algorithm used to hash the KRL RR, this hash is placed in the *Digest* field. For instance, the SHA1 [10] digest length is currently 20 bytes, so with SHA1 as digest algorithm, the LA RR size is 24 bytes. With the identifier and the digest, a signed LA RR authenticates a KRL RR of the child zone.

The LA RR associated with the revocation list is created by the child zone and then sent to the parent zone. This data exchange must be secure to avoid that a malicious person sends fake data to the parent zone.

We must provide integrity and authentication of the messages exchanged between the child zone and the parent zone. The message sent by the child to the parent must contain the LA RR. To protect this message, we use the SIG(0) mechanism [9]. A SIG(0) pseudo-RR (*pseudo* because this RR is never stored by DNSSEC clients or cache server and has a TTL of only a few seconds) contains a digital signature of the whole DNS message

(data and header). When a new revocation list is created, the child zone must sign its zone file. The private part of zone keys are available, we can use them to sign the message SIG(0) sent to the parent.

Our idea is to send the LA RR in a message signed by all KSKs of the child zone. An attacker can not create the fake message, unless he has compromised all the KSKs of the zone. Nevertheless, if this occurs, we fall in emergency risk management because there is no more safe chain of trust between the parent and the child. Administrators of parent and child zones must solve the problem together and as soon as possible. In this case, the new LA RR must be transmitted at this time.

### C. Using revocation records during a name resolution

Figure 5 shows a secure name resolution using KRL RR and LA RR.

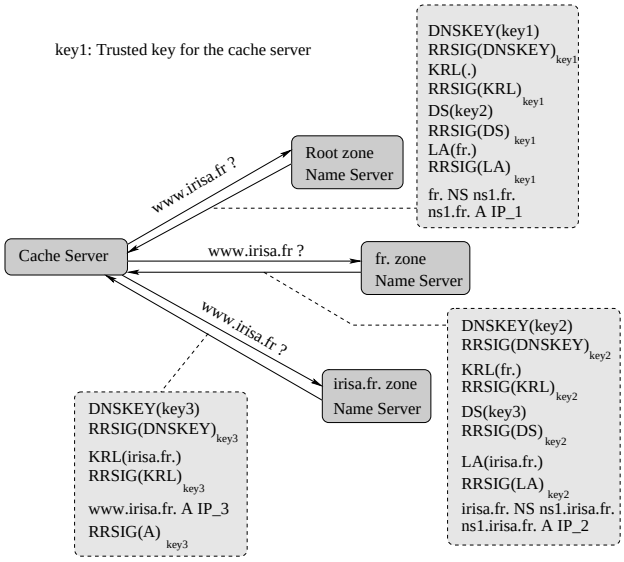



Fig. 5. Name resolution with compromised key check.

The cache server initiates the name resolution from one of its trusted key, the root zone key on this example. The first step is to obtain zone keys of the root zone, a signed KRL resource record and a delegation, DS RR and LA RR for the child zone and information about the following name server to query (IP address and name). The cache server checks that the keys it uses are not present in the revocation list.

Then, each time a zone is queried, the cache server obtains the zone keys and checks that it does not use any key in the revocation list. If such a key is found, this key is removed from the cache server and all RR signed by this compromised key are removed too. To trust a zone key the cache server must check that this zone key is not compromised.

On the example of Figure 5, the cache server receives the root zone keys, the KRL RR of the root zone, DS RR and LA resource record for the `fr.` zone. It verifies signatures and checks that the keys received are not in the KRL RR. Then, it sends a query to the `fr.` zone authoritative server, obtains the keys and KRL RR of the `fr.` zone, DS RR and LA RR for the `irisa.fr.` zone. It verifies signatures, checks that the LA RR previously received authenticates the KRL RR and then checks if some keys are compromised.

Finally, the cache server reaches the `irisa.fr.` zone. It obtains the keys, KRL RR and the queried resource. Once all checks are passed, the cache server stores resource records and sends the response to the resolver that previously sent the query.

### D. Effect of caching on name resolution

In the previous section, we have depicted a name resolution when the cache server asks directly the authoritative name servers and obtain fresh information. But, we have not always this optimal case. The cache server may have an old KRL RR and its associated LA RR in memory. Hence, a key that has just been compromised can be used to launch an attack. A way to mitigate this is to give to the KRL RR a short TTL, a few minutes for instance. Then, as the ID is ordered, if a cache server received RRs such as $LA.ListID < KRL.ListID$ with verified signature for the KRL and LA RRs, then a revocation process is underway, the DNSKEY must not be trusted (we assume that the cache server has detected the inconsistency and specifically asks for LA record to the authoritative name server to check if it has the most up to date LA RR). If $LA.ListID > KRL.ListID$, the cache server checks the freshness of the KRL asking the authoritative name server. If this KRL is the most up to date and as this inequality is impossible by definition, the DNSKEY must not be trusted. If $LA.ListID = KRL.ListID$ there is consistency between the two records, the name resolution can go forward (replay attacks are discussed in section IV). As the LA and the KRL RRs have a short TTL, if a key is compromised, the window vulnerability is also short.

## IV. Attacks and defenses analysis

We study now possible attacks against the revocation system. We assume an attacker can do anything (such as forge records and sign them with compromised a key) against DNSSEC. As the revocation records provides protection against compromised key attacks, we assume the attacker owns a compromised key. So, an attacker can perform two types of attack:

- cache pollution with forged records signed by the compromised key,
- old records replay to mislead a resolver or a cache server.

### A. Attack with forged records correctly signed

Figure 6 shows the previous name resolution example when an attacker try to put forged records in a cache server. This attacker has compromised a key (*key3*) of the `irisa.fr.` zone.

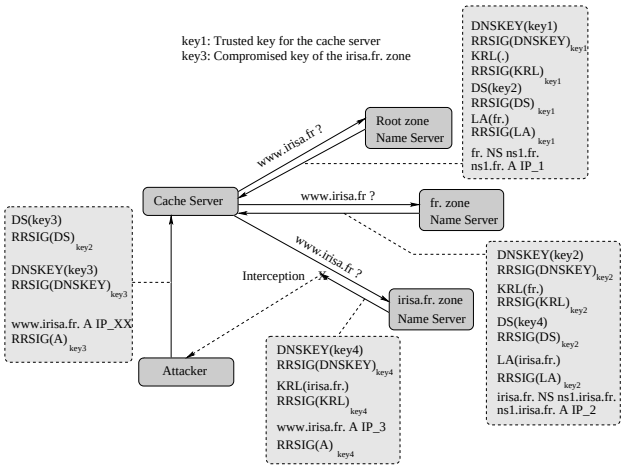



Fig. 6. Attack with compromised key and forged resource records.

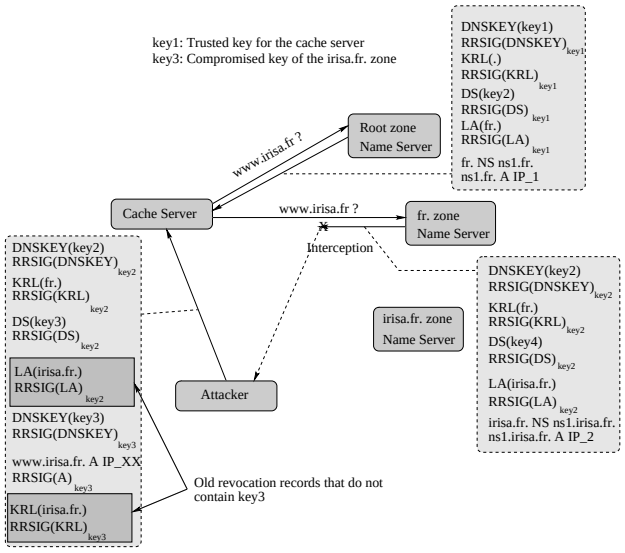



Fig. 7. Compromised key attack with old revocation list replay.

The cache server launches a name resolution following delegations after having verified the signatures, the revocation list and the link between LA and KRL RRs. The attacker having compromised a key of the `irisa.fr.` zone must prevent its target (the cache server) from getting the revocation list, with traffic interception for example.

As the KRL record of the `irisa.fr.` zone is missing. The cache server cannot check if the keys are compromised, so it sends one more query to get the KRL RR of the `irisa.fr.` zone.

- If the attacker intercepts the answer, the cache server can not finish the name resolution, it does not trust any RR received during this process and it notifies the resolver.
- If the cache server receives the KRL Resource Record of the `irisa.fr.` zone, it can check the revocation list and find the compromised key used by the attacker. Then, no RR are trusted and the cache server notifies the resolver.

In both cases, the attack fails. Using LA and KRL RRs prevents from compromised key attacks.

### *B. Attack with compromised key and old KRL RR replay*

When a cache server gets the most up to date KRL, it can detect a compromised key attack if the compromised key is in the KRL. To perform a compromised key attack, the attacker must bypass our revocation system. A simple KRL RR interception is not sufficient. The attacker must provide a correct and signed KRL RR that does not contain the compromised key. Then, it has to replay an old KRL RR with its associated LA RR. To obtain these records, it just has to previously send a query to the name servers.

Figure 7 shows a compromised key attack with old revocation list replay. We can see that all the records sent to the cache server allow to build a chain of trust authenticating the revocation list. Nevertheless, one or more revocation lists are not up to date. This is due to the static behavior of the DNS: when something is true at a time, this thing remains true until its signature expiration date.

Our revocation service must include an anti-replay mechanism. Using an identifier present in the zone or in the parent zone is not sufficient because all information previously published in the DNS may be replayed.

In the following, we analyze two approaches to defeat replay attacks. The first one is to separate the name resolution and the LA resource record distribution to sign it with the SIG(0) mechanism [9]. The second is to add LA RR in every DNS answer, which minimizes the time required to detect a replay attack.

### *C. Name resolution and SIG(0)*

Sending LA RR in a single message allows to manage it specifically. It is possible to protect the message containing LA RR with the SIG(0) mechanism. This specific behavior is done only on messages containing LA RRs. The signature needed to send the LA RR cannot be generated by the attacker, because the attacker haven't compromised the key of the parent zone. This is not the case if we adopt the same mechanism to sign the KRL RR, because the attacker owns a compromised key for the child zone.

The name resolution process is the same, the cache server simply asks for the LA RR on a separated query. When a name server receives a request about the LA RR, it signs the answer with the SIG(0) mechanism. An attacker can not replay an old message containing a LA RR. Indeed, the validity period of a SIG(0) signature is usually a few minutes and an attacker cannot keep an old message and replay it because of the expiration of the SIG(0) mechanism. If the attacker sends a message containing a LA RR without SIG(0), the cache server will drop this message because he can not prove its authenticity.

This method has two drawbacks, the first is to keep private keys of the zone on the name server to generate signature for the SIG(0) mechanism. The second is to use server resources to sign messages containing LA RRs.

### D. Another way to defeat replay attacks

As we have seen previously, using SIG(0) has some drawbacks. Another way to protect the revocation mechanism is to minimize the success rate of a replay attack. Indeed, such an attack succeeds if and only if an attacker provides DNS answers with old resource records before the authoritative name server. The attacker must provide an old version of LA and KRL RR. Two cases can arise:

- the target of the attacker already has one of both records (KRL or LA);
- the target of the attacker does not have revocation records.

In the first case, as revocation records have a serial number in the *List ID* field, a comparison between serial numbers (owned and received) is sufficient to decline the old resource record. The attack fails.

In the second case, we suggest to add the LA RR of the queried zone in every DNS responses and to give to this record a short TTL value. When an attacker wants to pollute a cache server with an old revocation list, it must intercept all DNS traffic from the parent zone to the polluted cache server. This is to prevent cache server to get more up to date information (higher serial number). Basically, two DNS queries on a given zone are needed to get all information (keys, resource records, signatures, *etc.*). This increase the number of reception of the LA RR and hence the probability of detecting an attack.

This solution has two advantages, it does not use servers resources and it does not need private keys on authoritative name servers to sign DNS messages with SIG(0).

## V. Conclusion

In this paper, we have seen that only one compromised key in a DNS zone threats all zones in the compromised zone's subtree. Even if a zone is secure without any compromised key, this zone can be threatened if one of its ancestor zone is compromised. We have analyzed consequences of a compromised key in DNSSEC and current defenses for this problem in [11]. This has shown that a revocation mechanism is needed in DNSSEC. Nevertheless, designing such a revocation system is not an easy task. Indeed, signatures remain correct until their expiration time. Hence, it is possible to build an old chain of trust if all the signatures in this chain have not expired. This raises the problem of replay attack in DNSSEC.

We have describe in this paper our revocation system for DNSSEC based on two new resource records: the Key Revocation List RR and the List Authentication RR. We have shown in Section IV that this mechanism resists to forged resource records attacks even if those records are correctly signed with a compromised key. We have also proposed two ways to defeat replay attacks against the revocation mechanism in Section IV-C and IV-D. The first approach is to sign DNS messages containing the LA RR with the SIG(0) mechanism. The second approach is to include the LA RR in every DNS response. This second approach is more scalable because it does not need additional computation time from authoritative servers.

Our revocation can be easily deployed to detect and defeat compromised key attack in DNSSEC.

## References


[1] Albitz, P., Liu, C.: DNS and BIND, 4th edn. O'Reilly & Associates, Inc., Sebastopol, Californie (2002)

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

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

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

[5] Atkins, D., Austein, R.: Threat Analysis of the Domain Name System. RFC 3833 (2004)

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

[7] Danzig, P., Obraczka, K., Kumar, A.: An Analysis of Wide Area Name Server Traffic: A Study of the Internet Domain Name System. In: ACM SIGCOMM 92 Conference, pp. 281–292. Baltimore, MD (1992)

[8] Eastlake, D.: Domain Name System Security Extensions. RFC 2535 (1999)

[9] Eastlake, D.: DNS Request and Transaction Signatures (SIG(0)s). RFC 2931 (2000)

[10] Federal Information Processing Standards (FIPS): Secure Hash Standards. Publication 180-1, U.S. DoC and NIST (1995)

[11] G. Guette: Consequences of compromised zone keys in DNSSEC. Research Report 5854, INRIA (2006). URL http://www.inria.fr/rrrt/rr-5854.html

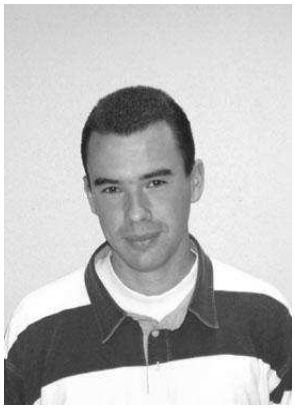
[12] Gieben, R.: Chain of trust: The parent-child and keyholder-keysigner relations and their communication in dnssec. Master's thesis, University of Nijmegen (2001)

[13] Gundmundsson, O.: Delegation Signer Resource Record. RFC 3658 (2003)

[14] Jung, J., Sit, E., Balakrishan, H., Morris, R.: DNS Performance and the Effectiveness of Caching. In: Proceedings of the ACM SIGCOMM Internet Measurement Workshop '01, pp. 153–167 (2001)

[15] Osterweil, E., Pappas, V., Massey, D., Zhang, L.: Zone State Revocation for DNSSEC. In: Proceeding of the 2007 workshop large scale attack defenses, pp. 153–160 (2007)

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


**Gilles Guette** received its M.Sc. in computer science and Ph.D. from the University of Rennes with honors in 2002 and 2005 respectively. After a post doctoral position at the university of Compiègne, working on the security of vehicular ad hoc network, it is currently associate professor at the University of Rennes. His current research interests include network security, key management and mobility.