Enhancing the Security of Chain of Trust in DNSSEC

Kin-Yeung Wong, Wei-Leung Koo, and Kai-Hau Yeung

Abstract— **DNSSEC provides origin authentication and data verification. It uses public key cryptography to build an chain of trust between parent and child name servers. There are two pairs of keys are used in DNSSEC, but only one of them, namely, KSK-Public is used to build the chain between name servers. This could cause a failure threat from a single point. That is, if an attacker could manage to compromise the ZSK-Public, it could modify the zone data and hence break the authenticate chain, making rogue servers and fake DNS response possible.**

In this paper, we propose a solution to enhance the security of chain of trust. It is to use double authentication process by making use of all the existing key pairs (KSK and ZSK). Since our solution is based on existing DNSSEC structures, it does not need to introduce any new key pair or any new DNS resource records, making it easy to be integrated into the existing systems.

Index Terms— **DNS, DNSSEC, Naming Systems**

I. Introduction

The Domain Name System (DNS) [1] is one of the fundamental building blocks of the Internet. It provides a mechanism for resolving human memorizable domain names into numeric IP addresses. However, similar to many Internet protocols, the original design of DNS protocol specification did not include security, which makes the DNS vulnerable to various kinds of attacks, such as cache poisoning and traffic diversion.

Since the proper functionality of the DNS is crucial to the Internet, the Internet Engineering Task Force (IETC) added a set of security extensions to the existing DNS protocol. The set of extensions is collectively known as DNS Security Extension (DNSSEC) [2],[3].

Although DNSSEC provides origin authentication and data integrity, the success of it relies on the correct operation of chain of trust, which is an authentication chain between name servers. The existing approach to build the chain of trust is to use the Zone Signing Key (ZSK) pair (one of the two key pairs used in DNSSEC).

In section 2, we first review some DNSSEC basics, which set a foundation for section 3 where we point out that the existing approach will cause a single point of threat. We show that it is possible for an attacker to compromise the ZSK pair. After that, it could modify and re-sign the zone data. In this case, the chain of trust would be broken, which allows the attacker to run its rogue name server and return fake DNS response to DNS resolvers. In section 4, we propose a solution to enhance the security of the chain of trust. This is to use of all the existing key pairs (KSK and ZSK) in DNSSEC. Advantages and disadvantages of our proposal are also included.

Manuscript received November 28, 2007. The work described in this paper is supported by Macao Science and Technology Development Fund (Project No. 099/2005/A).

K. Y. Wong is with the Computer Studies Program, Macao Polytechnic Institute (phone: +853-5996440; fax: +853-719227; e-mail: kywong@ipm.edu.mo).

W. L. Koo is a graduate from City University of Hong Kong (email: barrykoo@gmail.com).

K. H. Yeung is with the Department of Electronic Engineering, City University of Hong Kong (email: eeayeung@cityu.edu.hk).

II. DNSSEC Basic

We first review some DNSSEC basics, which set a foundation for the presentation of the DNSSEC security threat and our proposed solution.

A. Key Signing Key (KSK) and Zone Signing Key (ZSK) Pairs

DNSSEC provides origin authentication and data integrity, which is based on public-key cryptography. DNSSEC uses two key pairs: Key Signing Key (KSK) and Zone Signing Key (ZSK) pairs. Each key pair has its own public key and private key, totally four keys in one DNS server. They are named KSK-Public, KSK-Private, ZSK-Public, and ZSK-Private.

The ZSK pair is used to sign the zone's data. A name server uses its ZSK-Private to sign (encrypt) its zone data, and makes its ZSK-Public public so that others can use the key to decrypt the signed zone data (to check data integrity).

On the other hand, the KSK pair is mainly for authentication purpose. A name server sends its KSK-Public key to its parent DNS server. This is to build up the authentication relationship between parent and child name server. This relationship is called *chain of trust*, and will be discussed later. The child name server will also make its KSK-Public public so that others can check this against the copy in its parent server.

B. New Resource Records

To provide the security functions, DNSSEC adds four new resource records (RR), as shown in Table 1. The DNSKEY, RRSIG, and NSEC resource records are automatically generated in the zone signing process. More attention should be placed on the DS resource record. It is optional in DNSSEC and it will not be automatically generated. To generate the DS record, the DNS server has to first obtain its child's public key first. The DS record is particularly important for DNS servers, if they want to build the trust of each other.

Table 1. Resource records added in DNSSEC

DNSKEY	Storing public key string
RRSIG	Storing digital signature
NSEC	Storing next domain name
DS	Storing the hash of child's KSK-Public

C. *Chain of trust*

A name server's public keys (KSK-Public and ZSK-Public) are shown in the DNSKEY resource records of its zone file.

When a resolver wants to verify its received data, *D*, from a name server, it can perform a data integrity check. First, it gets the ZSK-Public key (as included in the DNSKEY record) from the server's response, and then uses the key to decrypt the signature (as included in the RRSIG record) to obtain the hash value of the original data, *H1*. Then, the resolver passes the received data, *D*, to the same hash function as the server use to calculate another hash, *H2*. After that, the resolver checks *H2* against *H1*. If they match, the received data, *D*, is verified and can be trusted.

However, there is a threat in the above verification process [4, 5]. That is, the public key in the first step has not been verified and can be incorrect. Therefore, if an attacker could manage to compromise a DNS server, it could modify the zone data, and also generate a new key pair to re-sign the zone data again, then publish its public key to all DNS servers or clients. In this case, resolvers would not be aware of the problem, since they use the attacker's newly generated public key to verify its forged data, and hence, there would be no error during the authentication process.

To solve the problem, chain of trust is used to build up an authenticated relationship between parent and child DNS servers. It requires that a parent server to verify the public key of its child server through the use of DS resource record.

To build the chain:

- A name server (e.g., example.com server) first places its KSK-Public in the DNSKEY record of its zone file. (It also places its ZSK-Public for data verifying purpose).
- It then sends the KSK-Public to its parent name server (i.e., .com server) in a secure way. The "way" is an operational matter (could be done via email) and not covered in the DNSSEC specification.
- When its parent received the key, it will generate a DS record to store the hash of the child's KSK-Public in its own zone file.
- The parent also needs to sign the DS record using its own private key (ZSK-Private) and store the result in a RRSIG record.

The zone file in this example is shown in Fig. 1.

To authenticate an origin, say, example.com

- All the public keys have to be stored in its zone file, so there are two DNSKEY records.
- One first uses KSK-Public$_{example.com}$ (as shown in the DNSKEY record) to decrypt the RRSIG DNSKEY RR (i.e., (...*bbb*...) in Fig. 1) to obtain the hash value of the KSK-Public$_{example.com}$, say, *H1*.
- It then passes the KSK-Public$_{example.com}$ to the same hash function to generate another hash value, say, *H2*.
- By comparing *H1* against *H2*, if they match, the integrity of KSK-Public$_{example.com}$ is verified.
- After verifying the key, the resolver then checks *H2* against the hash stored in the parent's DS record.
- If they match, the child name server is authenticated.

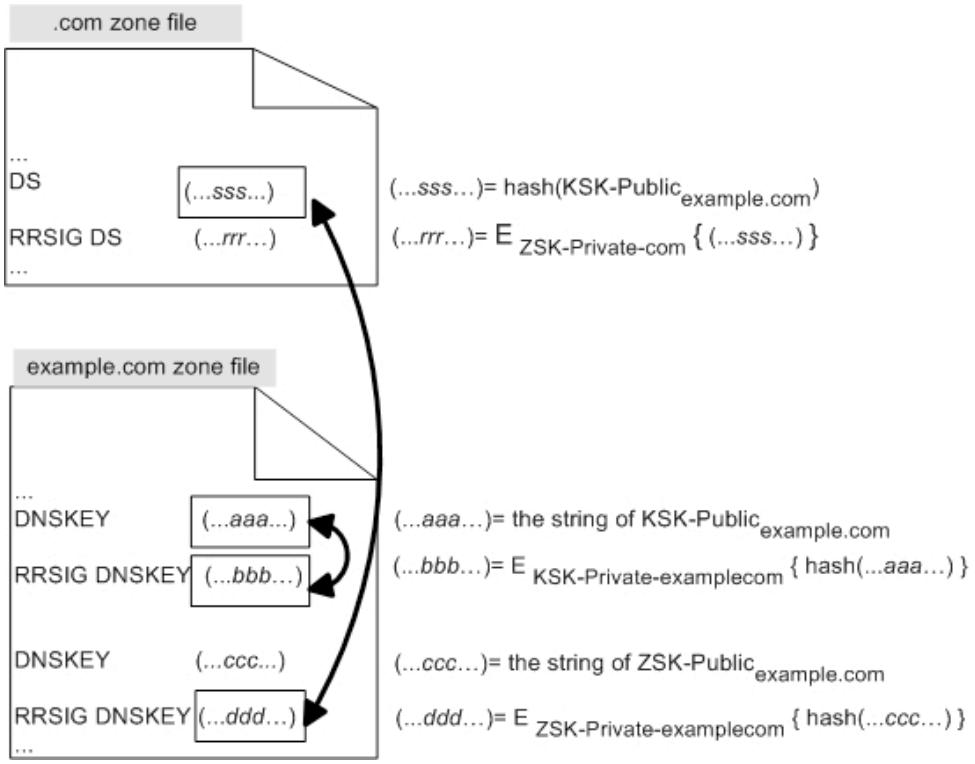

Fig. 1. Origin authentication.

III. Security Threat to Chain of Trust

As can be seen, chain of trust provides authentication through the use of the parent's DS record and the child's DNSKEY record. However, as the child use its KSK-Private to sign its DNSKEY record which stores the KSK-Public, and use its ZSK-Private to sign all other zone data including the DS record, this may cause a single point of vulnerability.

For example, if an attacker could manage to obtain (later, we will show some possible ways) a parent server's ZSK pair, then it would have the ability to modify and re-sign the zone data including the DS records. That is, the attacker can modify the DS record to store a malicious KSK-Public (which is another child server running by the attacker) and re-sign it using the compromised ZSK pair. This would cause a serious impact. It is because this alters the authentication chain and hence breaks the chain of trust. As a result, the parent server will refer all the resolvers to the malicious child server.

Fig. 2 shows an example to illustrate the threat. In normal case (see Fig. 2(a)), to resolve the address of host.example.com, the correct resolving path is from ns.root, ns.com, to ns.example.com name servers. In com zone, the DS record stores the hash value of example.com's KSK-Public. The com name server (ns.com) will tell the resolver that ns.example.com is the authoritative server, and send the resolver the hash value of example.com's KSK-Public (based on its DS record). The resolver then generates a new hash value using the KSK-Public from example.com's DNSKEY record. If these two hash values match, the resolver is confident that this is the right server for example.com zone. However, if the attacker is able to compromise the ZSK pair of com zone, it can modify the zone data and the DS record to store the public key of its own malicious machine, claiming that it is the key of the real example.com zone. Consequently, the resolving path would become from ns.root, ns.com, to the attacker's machine. See Fig. 2(b).

A. Compromising ZSK Key

In order To efficiently manage the zone data (mappings of domain name and IP), the DNS allows dynamic update [4, 6] function to add or delete resource records on demand. To allow dynamic update, ZSK-Private has to be kept in the name server so as to sign the modified zone data. This creates a chance for the attacker to "steal" the ZSK-Private.

Another way to crack the private key is to use key cryptanalysis if the attacker has enough computing power. According to cryptanalysis, the larger data set it has, the higher chance an attacker to crack the private key. It is unfortunate that ZSK-Private is used to encrypt quite a lot of zone data, unlike KSK-Private.

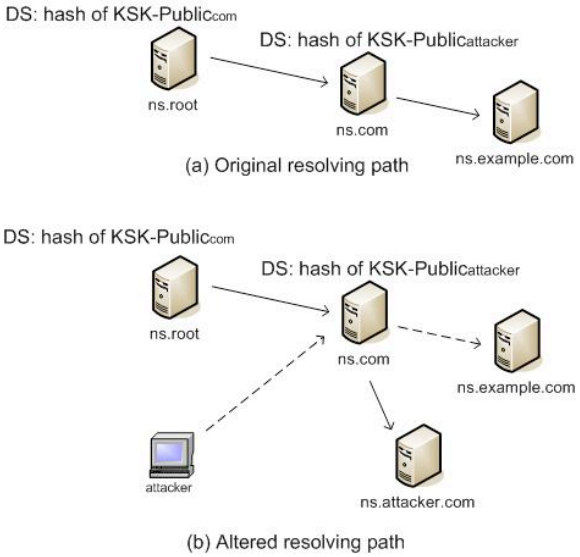

Fig. 2. Chain of trust is broken.

IV. Proposed Solution to Enhance Chain of Trust

In the currently DNSSEC specification, only one DS record is used to build the chain of trust. As mentioned before, this causes single point of security failure, and attackers with the ability to compromise the ZSK pair are able to break the chain of trust. To enhance the security of the authentication chain, we propose to use two DS records (instead of one record).

As each server has two key pairs: KSK and ZSK. In our proposal, the parent server will generate two DS resource records. One is used to store the hash value of its child's ZSK-Public, and its correspond RRSIG DS record is encrypted by the parent's KSK-Private. Another DS record stores the hash value of the child's KSK-Public and its RRSIG record is encrypted by the parent's ZSK-Private. Fig. 3 shows the comparison of the DS record in com zone between the original zone file and our proposed modified version.

Original approach
: DS: hash(KSK-Public$_{example.com}$) RRSIG DS: $E_{ZSK\text{-}Private\text{-}com}$ {hash (KSK-Public$_{example.com}$)} :
Proposed approach
: DS: hash(KSK-Publicexample.com)] RRSIG DS: $E_{ZSK\text{-}Private\text{-}com}$ {hash(KSK-Public$_{example.com}$)} : DS: hash(ZSK-Publicexample.com) RRSIG DS: $E_{KSK\text{-}Private\text{-}com}$ {hash(ZSK-Public$_{example.com}$)} :

Fig. 3. Sample zone files of example.com comparing the DS records between the original and our proposed approaches.

With our proposal, the chain of trust process would take two authentication checks. That is, both of the following conditions must meet:

The hash of a child's KSK-Public shown in the parent's DS record must match with the child's KSK-Public shown in the child's DNSKEY record.

The hash of a child's ZSK-Public shown in the parent's DS record must match with the child's ZSK-Public shown in the child's DNSKEY record.

When both cases pass, the public keys are said to be verified and can be trusted.

A. How the Proposed Solution Solve the Threats

It is noted that our proposed solution uses two different private keys (KSK-Public$_{com}$ and ZSK-Public$_{com}$) to encrypt the two DS RRs. The purpose of doing this is to add more security to the chain of trust.

In general, ZSK is less secure than KSK. It is because ZSK has to be stored in the computer for the dynamic update purpose, which leaves a security hole for an attacker to get it. However, since KSK needs not to be stored in the computer, and will be kept in secret, making it very difficult for attacker to crack. As a result, even though the attacker could manage to crack ZSK, if it could not crack KSK, it could not break the chain of trust. It makes our proposed chain of trust much more secure than the original one.

Another reason of why our proposal is secure is that KSK is very difficult to crack. As mention before, according to cryptanalysis, the larger data set (encrypted data), the higher chance an attacker to crack a key. Unlike ZSK which encrypts much zone data, KSK encrypts only the DNSSEC and DS RRs, providing only minimal data set for the attacker to crack KSK.

V. Conclusion

We proposed a solution to enhance the security of chain of trust through the use of two DS records storing both KSK and ZSK pairs. To deploy the solution, a minor modification to the existing DNSSEC specification is required (adding the DS records). It is based on existing key pairs, resource records, and authentication mechanism. Hence, it is feasible to be implemented.

However, the proposal will lengthen the authentication process because there are two key verification checks. Besides, if you change any one of key pairs, you have to inform the corresponding parent server for the change. As can be seen, the solution achieves higher security and robustness but causes higher processing and administrative overhead. Nonetheless, the primary concern of this work is security."

In the future, we plan to implement our proposal by writing our own zone-signing program which works with BIND [1].

References

[1] P. Albitz and C. Liu, *DNS and BIND,* 4th edition, O'Reilly, 2004
[2] R. Chandramouli and S. Rose, Secure Domain Name System (DNS) Deployment Guide, National Institute of Standards and Technology, USA, August 2005.
[3] O. Kolkman, DNSSEC HOWTO, Ripe NCC, April 2005.
[4] X. Wang, Y. Huang, Y. Desmedt, and D. Rine, "Enabling secure on-line DNS dynamic update," Proc. 16th Annual Conference Computer Security Applications, 11-15 Dec. 2000, pp.52-58.
[5] "Attacking the DNS Protocol – Security Paper v2," Security Associates Institute, 29 October 2003. http://www.rootsecure.net/content/downloads/pdf/sans_attacking_dns_protocol.pdf