**Sborník vědeckých prací Vysoké školy báňské - Technické univerzity Ostrava**
číslo 2, rok 2005, ročník LI, řada strojní
článek č. 1477

**Gert PFEIFER[*], Christof FETZER[**], Trevor JIM[***]**

# USING SSL FOR SECURE DOMAIN NAME SYSTEM (DNS) TRANSACTIONS

# VYUŽITÍ SSL PRO TRANSAKCE SYSTÉMU ZABEZPEČENÍ DOMÉNOVÝCH ADRES

**Abstract**

The main functionality of the Domain Name System (DNS) is to translate symbolic names into IP addresses. Since there is a growing demand for trustworthiness in the Internet many research articles consider DNS security issues. The reason is that DNS is a vital service for nearly all distributed applications based on the TCP/IP protocol suite. This article examines different possibilities to protect DNS transactions and shows an interesting alternative to the existing solutions. This is necessary since all existing approaches show significant flaws in terms of manageability, usability, scalability, versatility, and realizability.

**Abstrakt**

Hlavní funkcí systému doménových adres (Domain Name System - DNS) je překládat symbolické adresy do IP adres. Díky tomu, že se neustále zvyšují nároky na spolehlivost Internetu se mnoho odborných článků zabývá problémem DNS zabezpečení. Důvodem je, že DNS je životně důležitou službou pro téměř všechny distribuované aplikace založené na protokolu TCP/IP. Tento příspěvek se zaměřuje na různé možnosti jak chránit DNS transakce a ukazuje zajímavou alternativu již existujícím řešením. Je to důležité zvláště proto, že téměř všechny již existující přístupy odhalují významná mezery ve smyslu správy, používání, škálovatelnosti, univerzálnosti a implementace.

## 1 INTRODUCTION

The Domain Name System (DNS) is used to map symbolic names onto IP addresses. Many Internet applications rely on this service, which has been available for over 20 years. Nowadays approximately 300 Million hosts are using DNS [6]. Many of them are configured using technologies like the Dynamic Host Configuration Protocol (DHCP). This protocol is very often used in environments with mobile terminals like notebooks that are not permanently connected to the network. Due to the improvements in the construction of mobile devices like better displays, better batteries, and more powerful embedded processors they have an increasing popularity. The improvement of arithmetic capability enables an enormous number of small mobile devices like cellular phones to use TCP/IP and thus DNS. With this huge number of dynamically configured devices we must make sure, that the update mechanism of DNS information is scalable and reliable as well as secure.

During the last 7 years some techniques for dynamic DNS update, both secure and insecure, have been proposed and more or less established. This paper gives an overview and points out the typical problems of these approaches.

[*] Dipl.-Inf., Institute for System Architecture, Department of Computer Science, Dresden University of Technology, 01069 Dresden, tel. (+49) 351 463 39711, e-mail: gert.pfeifer@inf.tu-dresden.de
[**] Prof. Dr., Institute for System Architecture, Department of Computer Science, Dresden University of Technology, 01069 Dresden, tel. (+49) 351 463 39708, e-mail: christof.fetzer@inf.tu-dresden.de
[***] PhD., AT&T Labs Research, 180 Park Ave., Florham Park, NJ, 07932, USA, e-mail: trevor@research.att.com

The first attempt to dynamic update was made in the late 1990s. A message format was successfully established and is still used today and already at this point of time common sense said that there is a need for security mechanisms. Nevertheless security measures have not been standardized but just recommended. In the following years a couple of proposals have been made, including solutions included in secure DNS (DNSSEC). The most important of them are examined in this paper. Furthermore advantages and disadvantages of these solutions are examined and we propose an alternative that has noteworthy properties.

The most important design decisions are whether to prefer symmetric or asymmetric cryptography, whether to use regular DNS or DNSSEC as basis for the security measures, how to organize the management strategy, and whether to protect the transmission of DNS data by means of cryptography or the DNS data itself. All of those points have been proposed in different combinations leading to quite different approaches. The most important approaches are TSIG (Section 4), which uses manually configured shared secrets and the regular DNS to protect DNS transactions, and Secure Dynamic Update (Section 3) which is based on DNS security extensions that use asymmetric cryptography to protect whole DNS zones. This paper combines certain aspects of these very different approaches to come up with a solution which combines the most beneficial attributes of them without requiring changes to DNS and which uses existing infrastructures.

The remainder of the paper is organized as follows: Section 2 explains the ideas of the very first RFC on DNS dynamic update. Section 3 discusses an interesting DNSSEC-based approach and the Transaction Signature approach that is most commonly deployed today is introduced in Section 4. Section 5 combines the knowledge acquired so far and comes up with a new interesting approach. Section 6 concludes the paper.

## 2 DNS DYNAMIC UPDATE

Users should be able to update DNS entries without stopping the server, revising the zone file, and restarting the server. Therefore a dynamic update mechanism has been introduced in [9]. Server administrators are so able to allow, e.g., DHCP servers to update the zone file. The original DNS message format defined in [7] did not allow sending updates to the server. It was revised in [9].

A typical implementation strategy for this technique is to use a DHCP server that has the ability to send DNS update messages. Now each time when a new computer connects to the DHCP-managed network the DHCP server assigns an IP address to this computer and sends the mapping between the name of the computer and the new IP address to the DNS server. In most configurations like this the primary DNS server allows DNS updates from the DHCP server by checking the source IP address of the update message.

This method of authentication is very weak since it is vulnerable to spoofing attacks. The authors of [9] recommend not to use dynamic update without a suitable strong security measure.

The most important of these measures are examined in this paper and a new one is proposed.

## 3 SECURE DNS DYNAMIC UPDATE

Based on some extensions of the domain name system that have been defined to authenticate the data in DNS and provide key distribution services, [5] introduces public key mechanisms to authenticate DNS dynamic updates. These mechanisms are available on DNSSEC-aware DNS servers. DNSSEC uses digital signatures to authenticate data in the DNS database. In a signed zone file a SIG RR contains a digital signature for the RRs with the same owner name and class as the SIG RR. The name of the signer can be found in the signature. To verify the signature the public key, stored in a KEY RR, can be found by resolving the name of the signer. Since all RRs are signed in DNSSEC there is also a SIG RR provided for the KEY RR. In this manner the chaining through keys and signatures continues up to the root zone. The key of the root zone is a secure entry point for the verification process. In other words: everyone trusts the root key.

A dynamic secure zone contains one or more keys that can authorize dynamic updates. Two modes for dynamic updates are provided in [5]. Mode A is the maximum security mode. The zone owner key and the static zone file are always kept offline. There is a separate zone file that incorporates entries that have been received by update messages. These entries are signed with authorizing dynamic update keys. The server only checks whether these signatures are valid. The problem in this mode is that the NXT[1] RRs and the AXFR SIG[2] are not revised for any dynamic update and hence do not cover dynamically added data. Authentication for the server denial of the existence of dynamically added types or RRs is also not provided. Zone transfer security is also not automatically provided. Mode B is more powerful but less secure. The zone owner's key as well as the master zone file are kept online. When an authorized dynamic update is received by the server, the server calculates the necessary SIG and NXT records and incorporates the new information in the zone file. It is also able to calculate the AXFR SIG on demand, i.e., when a zone transfer is requested. If both modes are compared it is obvious that both have their disadvantages: While mode B is better able to provide secure up-to-date information including dynamically added data it also needs more computational effort than mode A and contains the risk that the zone owner's private key gets abused since it is online. Obviously mode A makes sense in zones with very static data and mode B is better suitable for zones with frequently updated data. But both of them suffer from some serious problems that come with DNSSEC, e.g., the hierarchical trust model, the roll-out consensus problem, the fact that DNSSEC servers are more effective as denial of service amplifiers, and the problematic root zone key roll-over.

## 4 TSIG

To provide transaction level authentication for DNS updates a simple but efficient protocol has been introduced by [8] in May 2000. The document specifies the use of message authentication codes (MACs) to provide authenticated point-to-point communication. It does not only focus on the DHCP server - DNS server relationship but also on general server to server communication including zone transfers. The protocol is based on manually configured shared secrets used to authenticate DNS update requests. The main argument for this solution is that public key cryptography is impractical for authenticating dynamic update requests because of the increased computational effort. So [8] is a lightweight alternative to [5] that has been discussed in Section 3.

The shared secret aka. secret key is used to calculate a RR type named TSIG. It is calculated dynamically on demand and not stored in the regular DNS manner. To calculate the message authentication code (MAC) the secret key and the one way hash function HMAC-MD5 are used. The TSIG contains the name of the algorithm, a timestamp, the MAC, and the data. When a DNS message is transmitted the TSIG is written into the ADDITIONAL section. Only one TSIG per message is allowed and it must be the last entry in the ADDITIONAL section.

The problem of this approach is that the number of shared secrets grows quadratically with the number of communication partners. The authors of [8] claim that the solution is suitable for many resolvers on hosts that only talk to a few recursive servers. This statement refers obviously to the relationship between stub resolvers and local name servers. Whether this makes sense is questionable.

Fig. 2 depicts how a DNS request is usually processed. The stub resolver sends a recursive request to the local DNS server. This server is responsible for resolving the name iteratively since the stub resolver is too limited for this task. To resolve the name the DNS server first tries to find the data in the cache. If this operation fails, many other DNS servers are involved to resolve the name. Each involved DNS server could try to send bogus information to poison the cache of the local DNS server or other adversary could manipulate the messages between servers. If only the connection between

[1] NXT (next) resource records - used to securely indicate that RRs with an owner name in a certain name interval do <u>not</u> exist in a zone

[2] signature covering the whole signed zone – used for security of zone transfers

the local DNS server and the stub resolver is protected using TSIG, it is most likely to be useless. The only sense would be to prevent manipulation between local DNS server and stub resolver. But this connection is easier to protect than the connections between DNS servers in the Internet since it is usually, e.g., within a building, perhaps within a VPN or at least within an administrative realm that the user knows and is able to influence.

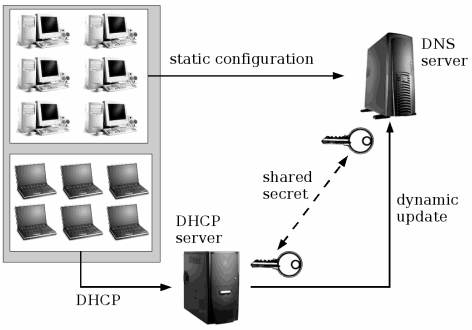


**Fig. 1** DHCP-DNS-Interaction

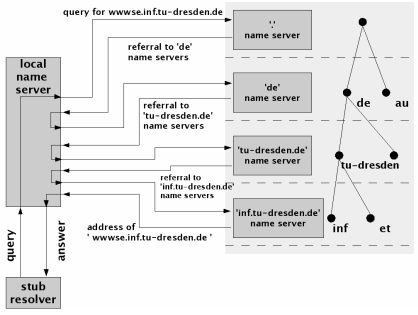


**Fig. 2** DNS-Resolving process

On the other hand it would be difficult to protect the communication between DNS server in the Internet using shared secrets. There are approximately 95,000 DNS servers available via the Internet [6]. Probably the overall number is much higher including all caching-only servers and servers that are not reachable because they are behind firewalls. The number of shared secrets would be enormous. So this approach cannot be usable for this purpose.

As a matter of fact TSIG is commonly deployed for the communication between DHCP servers and local DNS servers since those are often administrated by the same people and so it is quite easy to distribute and configure shared secrets manually. Fig. 1 shows a configuration that is often used in this context. A computing facility provides the ability to configure mobile hosts dynamically. These hosts need DHCP clients to connect to the DHCP server. This server assigns an IP address and provides additional information like DNS server addresses or the standard IP gateway address. It is also responsible for the registration of the assigned IP address on the DNS server. The DNS server allows updates based on TSIG keys.

Nowadays the trend is that administrators are generally using DHCP. Obviously it is a time saving technology. Since all hosts including many servers are accessed by names, the security of the DNS update mechanism gets even more important.

## 5 IMPORTANT CONSTRAINTS

Due to the experiences made with the solutions shown in Section 3 including the improvements of [3] and Section 4 we are now able to evaluate the advantages and disadvantages of these approaches and to define the criteria for our approach as follows:

**Not dependent on DNSSEC.** We do not want to suffer from any DNSSEC problems. The DNSSEC approach is now more than 5 years old [2] and the work on DNSSEC has begun more than 10 years ago. The roll-out is still bared by some serious problems so we want to use existing infrastructures like DNS and SSL to be able to deploy the service quickly.

**Coexistence with legacy software.** This point is a direct consequence of the use of existing technologies and infrastructures. The standard DNS protocol including the message format must not be changed since this would limit the interoperability with legacy products.

**Simplicity.** Since manual configuration is complicated or slow and automated configuration procedures often limit the flexibility and scalability, we try to make configuration as easy as possible. The configuration effort should not grow with the size of the system. Simplicity is also important for the trust model. We know that the complicated trust model of DNSSEC is not easy to use or

implement. We want to use a simple trust model with end-to-end semantics using certification authorities like it is used in connection with HTTPS.

**Usage of public key algorithms.** Public key algorithms provide some advantages in comparison to symmetric algorithms that we would like to use. They solve the key exchange and trust problems of symmetric cryptography.

**Usability for regular requests.** Our approach should be usable for dynamic updates as well as for regular DNS requests.

**Scalability.** Since DNS is a highly distributed database, scalability is a vital property. It bars TSIG from being commonly used for DNS requests.

### 5.1 Using SSL

SSL is a tunnelling protocol that was developed by Netscape in the mid-90s. It is used to authenticate both endpoints of a communication and supports encryption. Nowadays SSL is a common technology first and foremost in connection with HTTP. In this area it is a kind of a brand. People know what it means when there is a little yellow lock in the status bar of their web browsers. SSL is also often used to protect connections between e-mail clients and servers or to set up VPNs.

We would like to use SSL to protect the communication between DNS servers. The main advantages of this approach are that we can use the SSL public key infrastructure, therefore have a simple and successful trust model, have much library support in different modern programming languages at our disposal, and are able to use freeware open-source products to implement the SSL tunnel between DNS servers. In this approach DNS servers are able to check the integrity of DNS update messages using the public key of the updater. This public key can be obtained by using an X.509 public key infrastructure where each participant stores his own certificate containing the public key. In this case it would be suitable if certificates were available via the DNS itself like described in [4]. In our approach it is necessary to have a trusted third party, i.e., a Certification Authority (CA). It is recommendable to have more than one CA since CAs usually charge some money for a certificate. Competition between CAs helps to keep the prices low. The whole idea is comparable to HTTPS, which is very successful. If a web browser supports HTTPS it has its own little repository of trusted CA certificates. So it is able to check server certificates that have been signed by one of the trusted CAs and so to trust them. The same principle is suitable for DNS transactions as well.

Each DNS server needs a repository of trusted CA certificates to decide whether it can trust another server or not. If server A wants to transfer a DNS message to server B it can use its private key to sign the message. Server B can obtain and validate the certificate of A. This certificate contains the public key of A which can be used to check signatures made with the corresponding private key. So B can accept the DNS message if the certificate of A is signed by a trusted CA and is the signature check succeeds.

Another service that is provided by SSL is application layer encryption. It could be useful to encrypt DNS messages if you do not want to allow malicious programs to detect which host is allowed to update the zone file. As mentioned before we can use an existing product to provide this service, e.g., stunnel. Furthermore we need an SSL implementation like OpenSSL. Using these products we can set up SSL tunnels between non-SSL aware daemons without requiring any changes to these applications. If, e.g., DHCP server A wants to update the zone file information on DNS server B it can use it's certificate to set up an SSL tunnel between A and B. B checks A's certificate to decide whether it allows the tunnel to be established or not. Once the tunnel is ready, A can encrypt the DNS update message using the public key of B that is contained in B's certificate and send it through the tunnel. B is the only entity that is able to decrypt this message. Since DNS updates are rather rare, the tunnel may be closed after transmitting the message.

Using this algorithm, malicious programs are not able to detect which hosts are authorized to update the zone information and are not likely to steal their key. To make this even more unlikely one

could encrypt all DNS requests. Therefore the DNS servers would need more CPU power leading to a trade off between cost and security.

A problem that has not been discussed so far is the communication between stub resolvers and local DNS servers (Fig. 2). Stub resolvers are usually very limited library functions directly compiled into distributed applications. They are not able to do things like authentication and encryption using asymmetric cryptography. In this case it is recommendable to protect this connection by means like VPN, firewalls, or using the symmetric approach discussed in Section 4.

## 6 CONCLUSIONS

The approach presented in this paper shows how the most important advantages of the different proposals for enhanced DNS transaction security can be reached using existing infrastructures and technologies. In comparison to TSIG (Section 4) it provides the advantages of asymmetric cryptography and reduces the configuration effort. Once a list of trusted CA certificates and a list of update authorizing certificates are available there is no more configuration work. Exchange of this information is not as difficult as the exchange of secret keys, additionally there is the possibility to store these lists in the DNS itself which is not possible with secret keys. In comparison to the DNSSEC approach (Section 3) our approach does not depend on the roll-out of DNSSEC and is fully operational with regular DNS. Furthermore the trust model is less complicated. This is important since an important argument against asymmetric cryptography is the computational effort is too high. So it is obviously better if the signature validation chain is as short as possible. In our approach just the signature and the certificate of the signer have to be evaluated. To verify a signature in DNSSEC is more complicated. In addition to the authentication abilities, the SSL based approach provides encryption if wanted. This could help to prevent an adversary finding out which hosts are authorized to update DNS information and so prevents these hosts from being attacked more frequently.

**REFERENCES**


[1] ATKINS, D. & AUSTEIN, R. *Threat analysis of the domain name system.* Network Working Group, 2004.

[2] EASTLAKE, D. E. *Domain Name System Security Extensions.*RFC 2535, 1999.

[3] EASTLAKE, D. E. *DNS Request and Transaction Signatures (SIG(0)s).* RFC 2931, 2000.

[4] EASTLAKE, D. E. & GUDMUNDSSON, O. *Storing Certificates in the Domain Name System DNS.* RFC 2538, 1999.

[5] GUDMUNDSSON, O., KAUFMAN, C., KWAN, S. & LEWIS, E. *Secure Domain Name System Dynamic Update.* RFC 2137, 1997.

[6] INTERNET SYSTEMS CONSORTIUM. *ISC Internet Domain Survey.* http://www.isc.org/index.pl?/ops/ds/, 2005.

[7] MOCKAPETRIS, P. *Domain names - implementation and specification.* RFC 1035, 1987.

[8] VIXIE, P., GUDMUNDSSONAND, O., EASTLAKE, D. E. & WELLINGTON, B. *Secret Key Transaction Authentication for DNS (TSIG).* RFC 2845, 2000.

[9] VIXIE, P., REKTHER, Y., THOMSON, S. & BOUND, J. *Dynamic Updates in the Domain Name System (DNS UPDATE).* RFC 2136, 1997.

[10] WANG, X., HUANG, Y., DESMEDT, Y. & RINE, D. *Enabling Secure On-line DNS Dynamic Update.* Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC'00), 2000.


**Reviewer:** Prof. Christopher Irgens, University of Strathclyde, United Kingdom