Server Certificates based on DNSSEC[⋆]

Audun Jøsang and Kashif Sana Dar

University of Oslo[⋆⋆]
josang@mn.uio.no and kashifd@ifi.uio.no

Abstract. Globally unique domain names and IP addresses that are provided in real time by the DNS (Domain Name System) represent the fundamental signposts for navigating the Internet and for locating remote hosts. It is therefore paradoxical that the traditional method for secure identification of remote hosts is not directly based on DNS, but on the browser PKI which is external to the trust structure of the DNS and thereby introduces new and complex trust problems. This paper argues that certificates for global host names must be issued through DNSSEC. According to this principle, certificates for domain names are issued by DNS registrars or DNS server organisations. This greatly simplifies the trust models for online authentication and significantly improve Internet security.

1 Introduction

The DNS (Domain Name System) is a distributed network of servers that invisibly translates domain names (e.g. www.uio.no) meaningful to humans into the numerical names (e.g. IPv4 address 129.240.8.200) for the purpose of uniquely locating and addressing networked devices globally.

Security threats against the DNS are many [2, 5], which reduces the assurance in DNS responses such as IP address translations from domain names. The technical solution to this problem is DNSSEC (DNS Security Extension) [1] which was designed to protect Internet resolvers (clients) from forged DNS data, e.g. due to DNS cache poisoning attacks. All answers received with DNSSEC are digitally signed. Through validation of the digital signature a DNS resolver gets assurance that the information received is identical (correct and complete) to the information on the authoritative DNS server, i.e. that the information has not been corrupted. While protecting the integrity of returned IP addresses is the immediate concern for many users, DNSSEC can protect other information too, and it has been suggested to use it to protect standard public-key certificates stored as CERT records [4], thereby making it possible to use DNSSEC to distribute such certificates. However, the scheme proposed in [4] does not exploit the potential of DNSSEC for direct certification of domain names and IP addresses.

In this paper we describe how DNSSEC can provide a secure foundation for the management of public-key identity certificates that bind public keys to domain names and/or IP addresses, which can be used in the same way as current server certificates used with TLS (Transport Layer Security) or software signing certificates. The current

[⋆] Appears in the proceedings of NordSec2011, Tallin, October 2011.

[⋆⋆] The work reported in this paper has been partially funded by UNIK.

browser PKI is used to distribute public keys associated with domain names, so that the entities can prove ownership of domain names to other participants. The browser PKI offers relatively weak security assurance due to its implementation based on online distribution of root public keys. It is a bad idea to distribute a cryptographic key through the same insecure channel that the key is supposed to protect, but the browser PKI does exactly that. In fact, our scheme would make obsolete the current browser PKI infrastructure and thereby remove this vulnerability.

2 PKI Structures

Secure key distribution is a major obstacle to practical use of cryptography. With traditional symmetric-key cryptography each pair of parties that want to set up a secure communication channel must first exchange cryptographic keys through a secure extra-protocol channel and thereby establish a direct trust relationship. Secure extra-protocol channels and direct trust relationships are typically expensive to set up and operate, so finding ways to reduce their number can lead to significant cost savings. The main purpose of a PKI is to simplify key distribution by theoretically reducing the number of secure extra-protocol channels needed. Indirect trust in public keys is then cryptographically derived from a single direct trust relationship between the relying party and the root CA (Certificate Authority). In that sense, a PKI allows trust to be propagated from where it initially exists to where it is subsequently needed [8]. A detailed illustration of the trust involved in a certificate chain is illustrated in Fig.1.

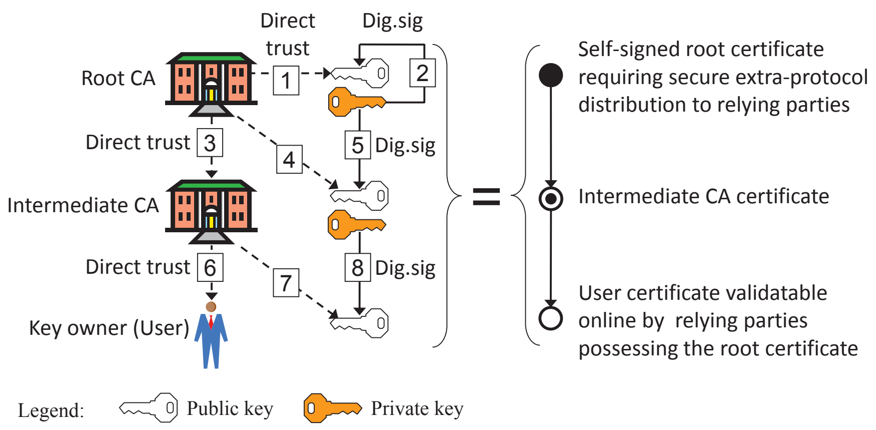
Direct trust
Dig.sig
Root CA
1
2
Direct trust 3
4
5 Dig.sig
Intermediate CA
Direct trust 6
7
8 Dig.sig
Key owner (User)
=
Self-signed root certificate requiring secure extra-protocol distribution to relying parties
Intermediate CA certificate
User certificate validatable online by relying parties possessing the root certificate
Legend:
Public key
Private key

Fig. 1. Detailed trust structure for certificate generation

The left hand side shows the trust structure where the indexes indicate the order in which the trust relationships and digital signatures are formed. The right hand side shows the corresponding graphical PKI representation. This trust structure can fail for several reason, typically resulting in false certificates being issued, as explained below.

Software systems are designed to store and process public-keys in the form of certificates, and are usually unable to handle naked public keys. For that reason a root public key is normally distributed and stored in the form of a certificate. The root certificate is normally self-signed, meaning that the public key sits in a certificate that has been signed by the corresponding private key, as illustrated at the top of Fig.1 (index 2). Note that self-signing by itself provides no assurance whatsoever regarding the authenticity of the root public key. Despite the fact that self-signing has no technical purpose many people falsely believe that it provide assurance and a basis for authenticity. In order to establish meaningful trust in root certificates the way in which root certificates are installed on a client, and the root CAs themselves should normally be known and trusted by users and relying parties. Most people ignore these issues and often download and install root certificates online even without knowing.

Validation of the certificate, normally done by the relying party, consists of verifying the digital signature on the certificate and extracting the data it contains, such as attribute/name and the public key. A detailed illustration of the validation procedure and the derived trust in the user's public key is illustrated in Fig.2. A relying party who holds an authentic copy of the root CA public key contained in a root certificate received through a secure extra-protocol channel, will be able to derive trust in the binding between the user public key and the user name.

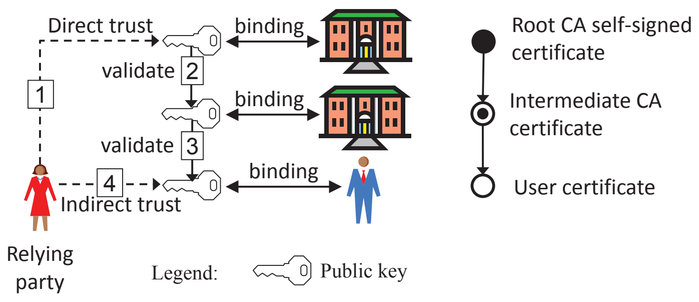

Fig. 2. Detailed trust structure for certificate validation

Recipients of public-key certificates, also called *relying parties*, do not themselves need certificates in order to authenticate a user's public key, they only need an authentic copy of the root public key. Only users that want to prove some attribute of themselves such owning a specific domain name, need to have public-key certificates.

The browser PKI used for secure Internet transactions and software installation consists of multiple hierarchical PKIs where each root certificate is stored in the browser, thereby enabling the browser to automatically validate any certificate issued under any one of the roots. Having multiple separate PKIs creates severe vulnerabilities through the fact the the whole browser PKI is only as secure as the weakest of each separate PKI. Thus, the more root certificates, the less secure the browser PKI becomes. The whole security chain would break if only one CA issues a certificate without properly checking owner's identity. This happened e.g. when VeriSign, the worlds largest CA, issued false certificates in the name of Microsoft, because Verisign failed to recognised that the persons buying the certificates were not Microsoft representatives [6]. The false certificates were never used and Verisign survived the breach with only a scratch to its reputation. The whole security chain would also break if only one private key were stolen. This happened e.g. when certificates were issued by DigiNotar's systems by an attacker with access to their systems. These certificates were used by criminals to conduct a man-in-the-middle attack against Google services [7]. A few months later DigiNotar was declared bankrupt. Browser PKI security is inherently weak because it depends on the security of the weakest of a relatively large number of root certificates.

3 Adjacent Structures of DNS and the Browser PKI

Single hierarchic PKIs can be operated by a single organisation that operates the root and multiple intermediate CAs, or by a set of separate organisations under one common root CA. EuroPKI is an example of the latter[1] model. However, the PKI commonly used for the Internet is a multi-root hierarchy where different user certificates belong to separate hierarchic PKIs, each with their own root. Assuming that each relying party shall be able to validate any user certificate from any PKI, then it is required that all root CAs represent trust anchors for the relying parties. In other words, all relying parties need to receive every root CA public key through a secure extra-protocol channel, but the browser PKI simply allows root certificates to be downloaded online, thereby making a mockery of the PKI security model.

Interestingly, the leaf nodes of the DNS are the same as those of the browser PKI, thereby making them adjacent hierarchic structures as illustrated in Fig.3 where the multi-hierarchic browser PKI at the bottom is turned upside-down.

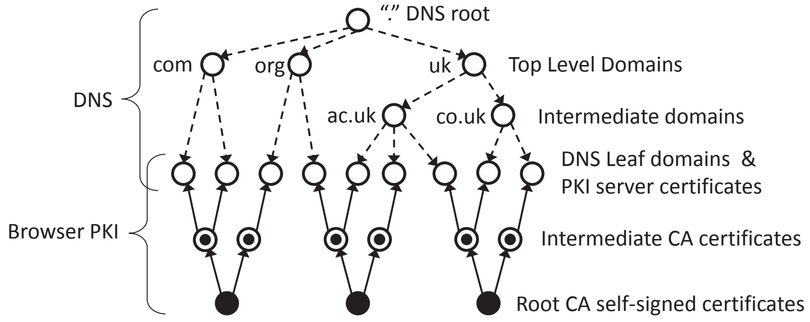

Fig. 3. Adjacent structure of DNS and the browser PKI

When authenticating a server on the Internet, the relying party depends not only on the integrity of the browser PKI but also on the integrity of the DNS. Should a relying party receive a corrupted reply to a DNS request e.g. in the form of a false IP

[1] http://www.europki.org/

address, then the relying party would be directed to the wrong server and the browser PKI in conjunction with TLS would still validate the wrong server to be authentic. The dependency between the DNS and the browser PKI is therefore conjunctive, i.e. both must function correctly for a server to be authenticated correctly. By expressing the reliability of the DNS as probability $P(\text{DNSSEC})$ (assuming that DNSSEC is used) and the reliability of the browser PKI and TLS as probability $P(\text{PKI})$, then the probability of a correctly authenticated server is their product expressed as:

$$P(\text{Server}) = P(\text{DNSSEC}) \times P(\text{PKI}) \ . \tag{1}$$

It is problematic that the correct authentication of a server has a conjunctive dependency on two separate systems because it reduces the overall reliability. The next section explains how server authentication can be simplified to depend on a single system while maintaining the same security functionality.

4 Using DNSSEC for Server Certificates

By looking at the diagram of Fig.3 it becomes obvious that the hierarchic structure of the DNS itself can be used as a PKI structure for user certificates. In fact DNSSEC is already an overlay PKI on top of the DNS making it possible for DNS resolvers (clients) to authenticate replies to DNS requests. What we propose here is an extension of the scheme proposed in [4] where traditional X.509 certificates belonging to the browser PKI illustrated at the bottom of Fig.3. We propose to let server certificates be signed by the DNS zone where the corresponding server is located, as illustrated in Fig.4 below.

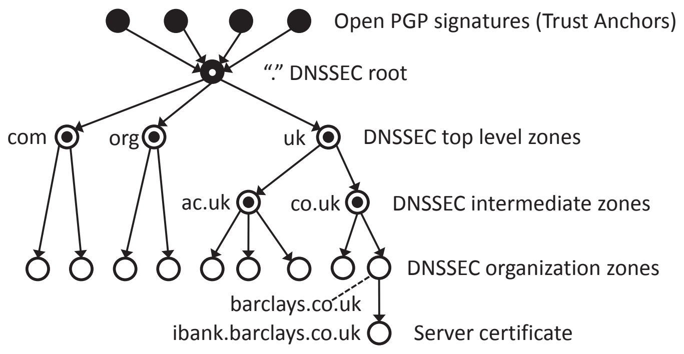

Fig. 4. DNSSEC as a platform for server certificates

In the example, Barclays bank's online banking server is called ibank.barclays.co.uk where the certificate is used for TLS connections. The public-key certificate for this server is signed by the public key of the DNS zone barclays.co.uk. The certificate can be stored as a RR (Resource Record) on the DNS server for barclays.co.uk so that it is available to all clients accessing the server.

In case of DNSSEC the trust structure is taken very seriously and multiple trust anchors represented by trusted individuals in the Internet community. Online validation of the DNS root public key is not possible, and is therefore called a DURZ (Deliberately Unvalidatable Root Zone). It does not mean that the DNS root public key can not be validated at all, instead the root public key can be manually (or semi-automatically) validated through the multiple OpenPGP signatures [3] on the root public key, as illustrated in Fig.4. So while the root public key associated with the "." DNS root can be downloaded online, its authenticity is based on some extra-protocol procedure. This can for example be that a DNS administrator obtains one or multiple OpenPGP public keys from people they trust, which in turn makes the DNS administrator able to manually validate the DNS root public key.

Integrating the PKI used for host authentication with DNSSEC results in host authentication only being dependent on the DNSSEC, so that the Eq.(1) is simplified to:

$$P(\text{Server}) = P(\text{DNSSEC})\ . \tag{2}$$

This solved the problem of depending on an separate trust structure in the form of the browser PKI which in addition must be characterised as relatively unreliable. Not only will the reliability of server authentication be strengthened, the cost can also be reduced because of the simplified infrastructure. When DNSSEC is deployed anyway it might well be used as a platform for signing and distributing server certificates.

5 Conclusion

This brief paper proposes to use DNSSEC instead of the browser PKI for signing and distributing server certificates. This solution leverages the strong trust structure of DNSSEC and thereby provides higher assurance of server authentication than is presently possible. In addition the proposed solution contributes to reducing cost because it creates a simplified PKI.

References

1. R. Arends et al. *RFC 4033 - DNS Security Introduction and Requirement*. IETF, March 2005. Available at: http://www.rfc-editor.org/.
2. Steven M. Bellovin. Using the domain name system for system break-ins. In *Proceedings of the Fifth Usenix Unix Security Symposium*, 1995.
3. J. Callas, L. Donnerhacke, Finney H., D. Shaw, and R. Thayer. *RFC 4880 - OpenPGP Message Format*. IETF, November 2007. Available at: http://www.rfc-editor.org/.
4. Simon Josefsson. *RFC 4398 - Storing Certificates in the Domain Name System (DNS)*. IETF, March 2006. Available at: http://www.rfc-editor.org/.
5. Dan Kaminsky. Details. Dan Kaminsky's blog at dankaminsky.com http://dankaminsky.com/2008/07/24/details/, 24 July 2008.
6. Microsoft. Microsoft Security Bulletin MS01-017 (March 22, 2001): Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard.
7. Elinor Mills. Fraudulent Google certificate points to Internet attack. http://news.cnet.com/, August 29 2011.
8. G.J. Simmons and C. Meadows. The role of trust in information integrity protocols. *Journal of Computer Security*, 3(1):71–84, 1995.