SSL/TLS PKI ISSUES

MARTIN RUBLIK

The most common network protocol that is used for encrypting web communications is SSL/TLS. During the last year we experienced several security incidents and this showed us that the risks associated with the management of cryptographic keys (not only in SSL/TLS) are real and relevant.

The motives for protecting the network communications by encrypting it are straightforward. It is quite easy to eavesdrop on network communications especially on wireless networks, or networks in a hostile environment controlled by an attacker. By eavesdropping on network communication an attacker can gain access to vital information. We use the communication networks for shopping, work and fun and we send quite a lot of sensitive information through them. If an attacker is able to access unencrypted/unprotected/plain communications, they can very easily impersonate the victim and gain access to victim's digital identity and accounts.

Researchers have come up with several means of protecting network communications using encryption. The purpose of encrypting the network communications is to preserve its attributes of confidentiality, availability and integrity. The most common network protocol for protecting the internet communications is SSL/TLS. SSL/TLS is used for protecting the web communications (HTTP/SSL), email communications (SMTP/SSL, IMAP/SSL, POP3/SSL) and also for protecting other kind of network communications like LDAP, FTP, RDP, XMPP/Jabber.

SSL/TLS protocol cryptography basics

The SSL/TLS protocol uses three kinds of cryptographic primitives to protect network communications. It uses symmetric encryption algorithms for protecting the confidentiality of communications, message authentication codes for protecting integrity and authenticity of the communications and asymmetric encryption algorithms for key exchange.

In symmetric encryption the sender converts/encrypts data/plaintext to encrypted data/ciphertext that is unreadable for an attacker. The recipient of the encrypted data decrypts the data to its original form and is able to read the data. To preserve the confidentiality of the data there must be some sort of information that is not known to the attacker. It can be either the algorithm that is used for encryption/decryption or an additional parameter that is passed to this algorithm. The encryption/decryption can be easily reverse engineered and thus does not constitute adequate protection, one needs an additional parameter that is called cryptographic key. This key needs to be safely transferred between the communicating parties so that the attacker cannot decrypt the information being transmitted.

Symmetric encryption is mostly used to protect the confidentiality of data. Though some symmetric encryption algorithms can be used also for protecting integrity and authenticity of data, in general this is not true. Therefore, if a symmetric encryption algorithm that does not protect the integrity and authenticity of data is used, one needs to use an additional cryptographic protection mechanism. Message authentication codes (MAC) can be used for this purpose in SSL/TLS network protocol. Basically MAC is a product of special one way cryptographic function (either based on symmetric encryption algorithm or hash function). This output can be used for verification of integrity of input data. Both communicating parties (creator of the MAC and its verifier) need to agree on the same cryptographic key that is used for creating and verifying a MAC.

Asymmetric encryption algorithms can be used for protecting confidentiality of data. Asymmetric encryption algorithms are based on different principles than symmetric encryption algorithms. The asymmetric encryption algorithm needs two cryptographic keys that are mathematically tied together. First key is used for encrypting the data and it can be available to everyone, especially sender of the data. This key is called a public key. Second key is called private key and is only available to the recipient of the data. This key is used for decrypting the data by the receiver.

The benefits that asymmetric encryption algorithms have over the symmetric encryption algorithms reside especially in simplified key distribution. It is not necessary to protect the confidentiality of public keys that are used for encryption of data. Unfortunately, asymmetric encryption algorithms are much more computationally demanding than symmetric encryption algorithms and are therefore used mostly for encrypting small amounts of data. One of such use cases is the symmetric key distribution. In this particular use case the symmetric encryption algorithm is used for protecting the actual data (making the encryption faster) and asymmetric encryption algorithm is used for exchange of symmetric keys (making the key distribution easier).

Though asymmetric encryption makes key management and distribution much easier, there are still several problems that need to be solved:

- integrity and authenticity of cryptographic keys,
- bonding between the cryptographic keys and their holder,
- validity and revocation of cryptographic keys.

These problems can be solved by several different approaches. First approach is out of band key distribution through a secure channel. This approach needs another secure channel in order to distribute a key (for example through personal interaction, USB, etc.) and can be used in circumstances where one communication party knows the other.

The second approach is accepting the public cryptographic key for the first time one party sees it (possibly verifying the integrity by using the band channel, like telephone or other means of personal interaction) and storing it for future use. This type of key exchange is used for example in SSH protocol.

The trouble with first two approaches is that they really do not scale well for large communities (e.g. web shops like Amazon) and it is quite hard to perform revocation of cryptographic keys. Amazon does not really want to prepare a call-center just for verifying the integrity and authenticity of cryptographic keys. Also this approach is not comfortable for an average user.

The third approach is based on a premise that there exists a trusted third party (TTP) that performs key certification. This TTP performs identity validation, and issues a statement where it bonds the public key and the identity of its holder. The statement is protected by cryptographic means with the key of the TTP. This way the problem of key distribution is reduced to distribution of one/several TTP cryptographic keys and providing a public key infrastructure (PKI) that solves the bonding between cryptographic keys and the identity of their holder as well as dealing with revocation of cryptographic keys. SSL/TLS protocol, how it is in use today, utilizes the third approach.

SSL/TLS PKI basics

SSL/TLS PKI is based upon X.509 public key certificates. In X.509 public key infrastructure the trusted third party is called a certification authority (CA). CAs issues the certificates in order to bind the holder's public key to a domain DNS name that is used for network communications.

The certificate has structure as defined in "ITU-T X.509 Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks". The general structure of certificate is illustrated in 1. A sample certificate can be viewed in browser by clicking the address bar padlock after SSL/TLS channel establishment.

The CA can either issue certificates to other CA's or directly for the users (also called end entities). The reason for having a CA hierarchy is to simplify the CA's key/certificate distribution. This way it is possible to build an infrastructure where there is only small amount of root CAs that issues certificates to several subordinate CAs and you need to distribute only the small number of the root CA certificates.

Every CA can have its own scope of issuing certificates and own set of policies that place requirements on operational procedures, security precautions and procedures for end entity identity verifications.

In TLS/SSL PKI there are three most common types of identity verification:

- domain validated,
- organization validated,
- extended validation.

The content of the public key certificate is based on identity validation process. Domain validated certificates contain in subject name (holder information) only the DNS domain name that is owned by the end entity (e.g. CN=*www.domain.org*). These certificates are easiest to obtain and are cheapest. The validation process is/can be done in an automatized way and involves mostly WHOIS checks or sending verification e-mails to a pre-configured e-mail address (such as *hostmaster@domain.org* or *postmaster@domain.org etc.*). The e-mails typically contain a one-time password that is used for identity validation during the certificate issuance process.

Organization validated (OV) certificates contain more information about the end entity. Besides the checks concerning the ownership of a domain, there are also other types of checks concerning vetting the organization. These checks can be based on public registers of organizations or based on scanned invoices. The process that is used for validation of OV certificates is semi-automated and could involve a personal contact between the certification authority officer and the end entity. The OV issued certificate contains in subject name the DNS domain that was checked as well as information about the organization that is the holder of domain (e.g. the subject could look like CN=*www.domain.org*, O=*Organization Inc.*, L=*Wien*, C=*AT*).

The third type of identity verification process is extended validation (EV). The EV issued certificates have similar verification process to OV certificates, and in addition, a more thorough/rigorous verification process of applicants. Furthermore, the process is standardized by CA browser forum. Some other technical requirements imposed on the certificate are the need for stronger cryptographic algorithms1 and stronger cryptographic keys2. Also EV protected network communications are indicated by internet browsers using a green address bar.

The third type of identity verification process is extended validation (EV). The EV issued certificates have similar verification process to OV certificates, and in addition, a more thorough/rigorous verification process of applicants. Furthermore, the process is standardized by CA browser forum. Some other technical requirements imposed on the certificate are the need for stronger cryptographic algorithms and stronger cryptographic keys2. Also EV protected network communications are indicated by internet browsers using a green address bar.

If the web site administrator wants to enable SSL/TLS protection of the network communication he needs to obtain an X.509 certificate from certification authority. There are several commercial certification authorities that issue X.509 certificates. The web site administrator should choose a CA that has its certificate shipped with major internet browsers. This way the user won't get any warning messages during the SSL/TLS channel setup and the network communication is less prone to eavesdropping.

If the web site administrator obtains a certificate from a CA that is not shipped with internet browser, the browser issues a warning to the user. The user should verify the authenticity of the certificate afterwards. However this is seldom done by users and it is really inconvenient. If the certificate is not verified by the user, an attacker could perform a man-in-the-middle attack where he would create a certificate with same name but issued by a CA that he can control.

After the web site administrator configures the web server, it can start to serve content over SSL/TLS protected channel.

How to attack SSL/TLS encrypted channel

There are several ways to attack an encrypted channel. The level of complexity of an attack depends on the level of sophistication of the process/IT security of its target. The attacker can choose to target the encryption algorithms (either from mathematical point of view, or from implementation's point of view), or attacker can choose to target the protocol (again either from design point of view or from implementation point of view), or an attacker can choose to target the key management and distribution infrastructure.

Though there are known attacks on encryption algorithms used in SSL/TLS protocol, these attacks are too far from being practical. These attacks are big topics for mathematicians but for internet users, system administrators and attackers as well, they are absolutely irrelevant.

There is a known attack against TLS 1.0 protocol implemented by Rizzo and Duong (BEAST attack), but it is still not easy to successfully execute this attack. The attack requires the user's interaction (e.g. a visit to a special crafted malicious web site), it also has some requriements on communication's behavior (e.g. the communication needs to have a secret, for example a cookie, on predictable location).

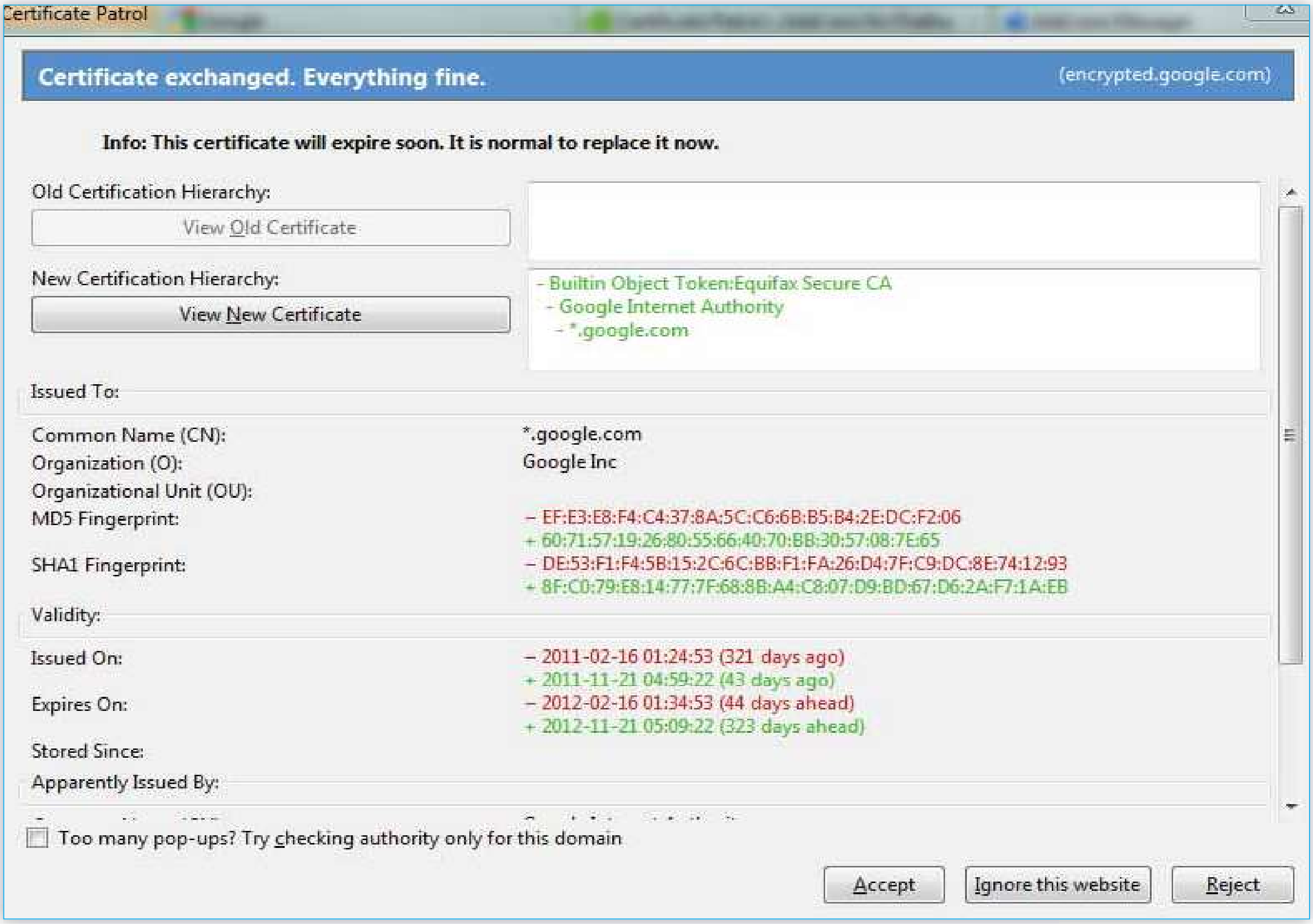
On the other hand, in the previous year alone, there were at least two known compromises to SSL/TLS public key infrastructure. As a result of these attacks Comodo CA and DigiNotar issued fradulent certificates to an attacker that did not control the specific DNS domains. This way attacker could mount a successful attack against any web site that uses SSL/TLS protocol for protection of network communications as long as he could control the network on lower layers. There are some indications that these fradulent certificates were used during an Iranian attack on public e-mail services such as Google Mail.

The attack is quite simple. During the first phase of SSL/TLS protocol the attacker performs a man in the middle attack by inserting a legitimate certificate from compromised CA. If the CA is not aware of being compromised and/or the attacker can block CRL/OCSP communication with CA, the browser cannot determine whether the certificate was fradulently issued and consequently, considers the certificate as valid. Afterwards the client will encrypt the symmetric key with attacker's public key and attacker can decrypt all the traffic designated for the web server. He can than act as an active proxy server in the communication.

There are several reasons why this attack is the easiest one. Their summary is as follows:

- revocation either by CRL or OCSP is by default fail-safe (meaning that if the browser is unable to retrieve revocation information the SSL/TLS channel setup proceeds, and sometimes it proceeds even without a warning),
- the users are not educated enough, meaning that the attacker does not even need to attack the CA. He can issue a certificate with a non-trusted CA and many users will just ignore the browser's warning,
- the web servers are misconfigured for SSL/TLS (either by serving non trusted certificate, or by serving a certificate with mismatched name, or by serving an expired certificate, or by using deprecated cryptographic algorithms, or by violating the SSL/TLS specifications),
- there are many (perhaps too many CAs) trusted by the internet browsers.

In 2010 the researchers from EFF presented results from their SSL Observatory project. The main objective of this project was to create a view on the SSL/TLS PKI by mapping publicly available web servers that serve content through HTTP over SSL. The results are very interesting. According to researchers there are more than 650 subordinate certification authorities that are trusted by major internet browsers. This number is biased as the root CAs are not obligated to publish the number of subordinate CAs to which they have issued a certificate. Therefore the researchers decided to estimate the number based on different name in CA certificate, especially different organization value. Another estimate, which can



Certificate exchanged. Everything fine. (encrypted.google.com)

Info: This certificate will expire soon. It is normal to replace it now.

Old Certification Hierarchy:

New Certification Hierarchy:

- Builtin Object Token:Equifax Secure CA
  - Google Internet Authority
    - *.google.com

Issued To:

Common Name (CN): *.google.com
Organization (O): Google Inc
Organizational Unit (OU):
MD5 Fingerprint: – EF:E3:E8:F4:C4:37:8A:5C:C6:6B:B5:B4:2E:DC:F2:06
+ 60:71:57:19:26:80:55:66:40:70:BB:30:57:08:7E:65
SHA1 Fingerprint: – DE:53:F1:F4:5B:15:2C:6C:BB:F1:FA:26:D4:7F:C9:DC:8E:74:12:93
+ 8F:C0:79:E8:14:77:7F:68:8B:A4:C8:07:D9:BD:67:D6:2A:F7:1A:EB

Validity:

Issued On: – 2011-02-16 01:24:53 (321 days ago)
+ 2011-11-21 04:59:22 (43 days ago)
Expires On: – 2012-02-16 01:34:53 (44 days ahead)
+ 2012-11-21 05:09:22 (323 days ahead)
Stored Since:

Apparently Issued By:

□ Too many pop-ups? Try checking authority only for this domain

Figure 1. *Certificate patrol information about certificate change*

be considered a lower bound, is 50 CAs (this estimate considers several CAs running at same location and operated by same personnel as one CA), but unfortunately the latter estimate does not have any proof whatsoever.

Also the Observatory project showed that CA practices are not followed rigorously. For example there were found several EV certificates that were issued to a non-qualified domain name (e.g. *webmail.domain.local* or *intranet*) or certificate issued to a short key (512 bit RSA).

Another research conducted by researchers from Technische Universität München also supports the conclusion that the SSL/TLS PKI is in sorry state. Their research was more thorough than EFF's and ran in longer time span, but concluded to similar results as the Observatory project.

Present and future countermeasures

The ugly part about the SSL/TLS PKI is that the attacker only needs to circumvent the safeguards of a *single* trusted certification authority and he is able to attack almost1 any internet site that uses SSL/TLS for protecting its communications.

With such high number of CAs (even lower bound is still high enough) the attack surface is attractive to attackers. It is hard to withdraw the certification authority from the browser trust store as it would break the established trust and it would mean that all the certificates that were issued through this CA needs to be replaced. Based on current sorry state of SSL/TLS PKI mis-configurations, one can assume that many server certificates would be omitted leading to user confusion and leaving the doors open to the attackers.

There is a lot of work to be done by browser vendors and CAs. Browser vendors should update their CA inclusion policies and should require that CA publish information regarding audit reports, security practices of root CAs and issue subordinate CAs as well and breach disclosure policies.

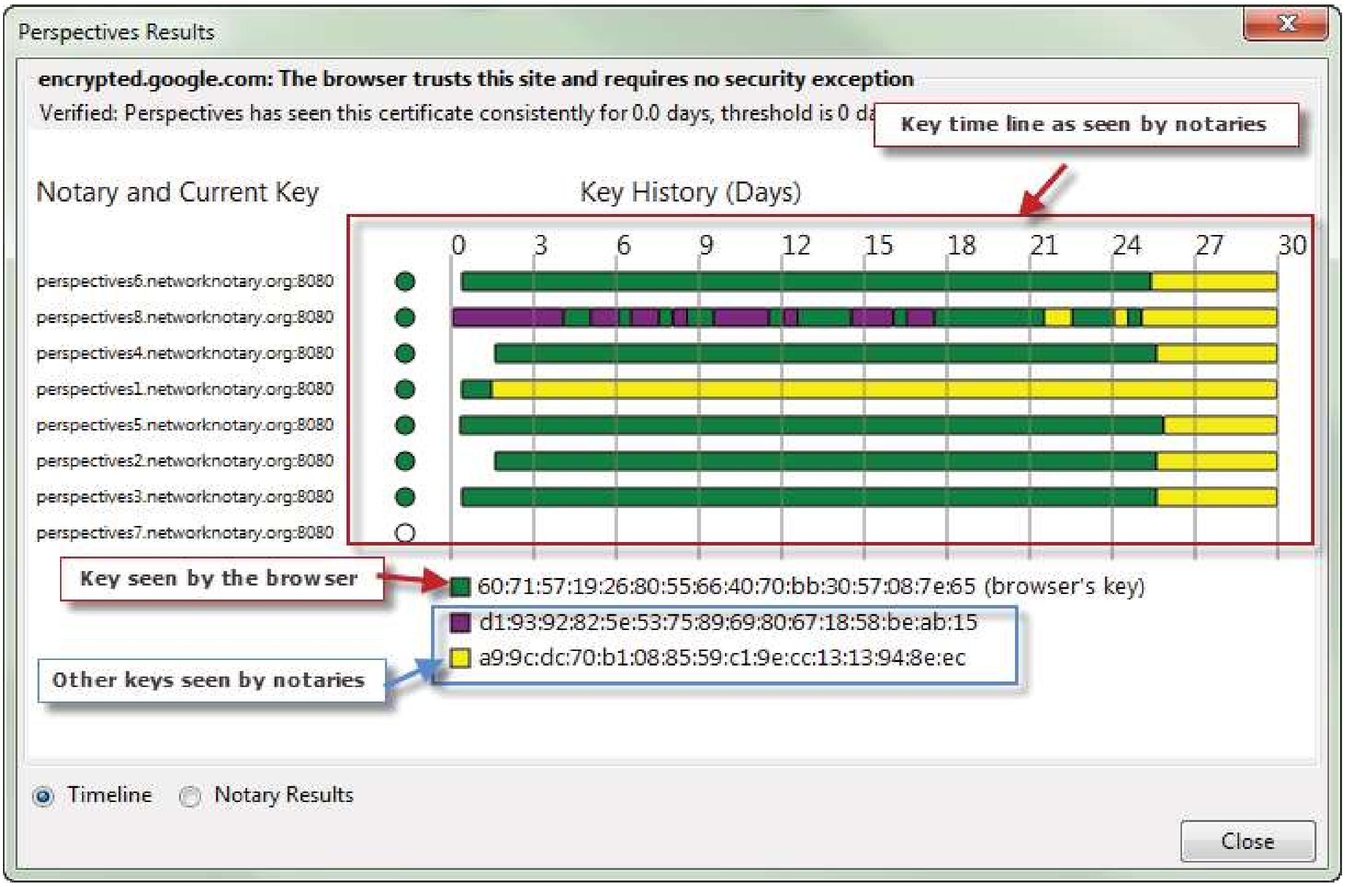
There is also a second possibility how to improve the security of SSL/TLS PKI. This possibility is based on a cross check of CA issued public key certificates.

The most simple way to perform a cross check is to use an SSH like key continuity principle in conjunction with traditional X.509 PKI. There is a Firefox extension called Certificate Patrol that supports this kind of protection. First time a user visits a web site he can verify the SSL/TLS certificate manually and Certificate Patrol stores it locally. If the Certificate Patrol discovers that a new certificate is being used, it will show the differences and alerts the user so he can make a decision whether he trusts the new certificate.

We have discussed this approach earlier, and also identified several problems (especially that it is not very comfortable for non tech-savvy users). Besides these issues there is one more problem. If the web site uses multiple certificates (for example because of load balancers), it will issue false positives.

Another approach was proposed by researchers from Carnegie Mellon University. Their solution (Perspectives project) is based on a separate public key infrastructure and a geographically distributed network of certificate notaries. These notaries scan internet sites for SSL/TLS servers and store the certificates they see. When the browser sets-up a SSL/TLS connection it'll contact several notary servers and compares the results. The Perspectives browser plugin can be configured on how many notaries need to see the certificate consistently and for how long.

This way the attacker would need to attack the network closest to the server and for a longer timeframe, so that every notary server views the fraudulent certificate over a longer period. This makes attacks significantly harder, even impossible, especially for


Perspectives Results

encrypted.google.com: The browser trusts this site and requires no security exception

Verified: Perspectives has seen this certificate consistently for 0.0 days, threshold is 0 da

Notary and Current Key

perspectives6.networknotary.org:8080
perspectives8.networknotary.org:8080
perspectives4.networknotary.org:8080
perspectives1.networknotary.org:8080
perspectives5.networknotary.org:8080
perspectives2.networknotary.org:8080
perspectives3.networknotary.org:8080
perspectives7.networknotary.org:8080

🟩 60:71:57:19:26:80:55:66:40:70:bb:30:57:08:7e:65 (browser's key)
🟪 d1:93:92:82:5e:53:75:89:69:80:67:18:58:be:ab:15
🟨 a9:9c:dc:70:b1:08:85:59:c1:9e:cc:13:13:94:8e:ec

Figure 2. Perspectives plugin certificate report

man in the middle attack scenarios where attacker controls only the local area network.

This solution does not have the drawbacks of the key continuity, as it is more user friendly and can deal with the key rotation/ multiple certificates problem (as illustrated by 4). It even deals with the problem that can be caused when a server certificate is issued by a non-trusted certification authority, because the verification is done independently from browser's set of trusted root certification authorities.

Moxie's Convergence project is also built upon Perspectives idea. This project is currently still in beta version and it supports the same logic for determining whether the certificate is considered valid or not. However in the future it plans to rely on multiple sources of information. These could include online notaries, EFF Observatory database, CA sources and DNSSEC based information.

Besides already presented solutions there is ongoing discussion on standards track. IETF PKIX working group and IETF DANE working group are preparing standards proposals that would help to mitigate the risks connected to SSL/TLS PKI.

DANE (DNS Based Authentication of Named Entities) provides bindings between the domain name and the server's public keys / certificates. The idea behind is to use DNS protocol to bind DNS name, port and cryptographic keys. DANE extends the DNS protocol by adding a new resource record type (TLSA) to the existing resource record types (HOST, PTR, CNAME, MX, etc.). The TLSA RR will bear several types of information.

First type of information specifies which of the certificates received by the client during SSL/TLS channel setup should be used during cross-check against TLSA RR. It can be either a CA certificate (either a root CA certificate or one of the subordinate CA certificates), or it can be directly server's certificate. This information determines how the information from TLSA RR should be used by a client.

Second type of information specifies the content of the TLSA record. This can be either information regarding public key only or it can be full public key certificate information.

Third type of information specifies how TLSA association is presented. This can be either:

- full information (e.g. public key information or X.509 certificate),
- hashed information – digital cryptographic thumbprint of full information.

An example of a TLSA record according to DANE protocol draft#13 is illustrated by 5.

The DANE strengths are similar to Perspectives. It can provide cross-check to CA issued certificates and it can also be used to trust certificates directly (without the necessity of including a CA in a browser trust list). It is also worthy to mention that DANE could be easily implemented in a non-browser environment that cannot rely on user decisions and interaction (e.g. MTA software for STMP over SSL/TLS).

Unfortunately DNS is not a secure protocol and therefore one needs to use DNSSEC, in order to provide protection for associations built by DANE. However DNSSEC also relies on a public key infrastructure where the trusted third parties are domain registrars. The improvement that comes with DNSSEC is that in case of one TTP compromise, only the part of the infrastructure for which the TTP was authoritative is in danger.

Similar approach was chosen by PKIX IETF working group. The PKIX WG proposed to extend DNS resource records with CAA (Certification Authority Authorization) RR. This resource record would be used by certification authorities as part of background checks prior issuing a certificate. This way the risk of unintended certificate issuance would be reduced. The resource record should contain information about a certification authority that is allowed to issue a certificate for a specific domain, as well as contact information that would be used by CA in case when CA is unauthorized to issue a certificate (based on CAA RR). This way the domain owner can be notified that someone is trying to acquire a certificate from another CA than the one that is indicated in CAA RR.

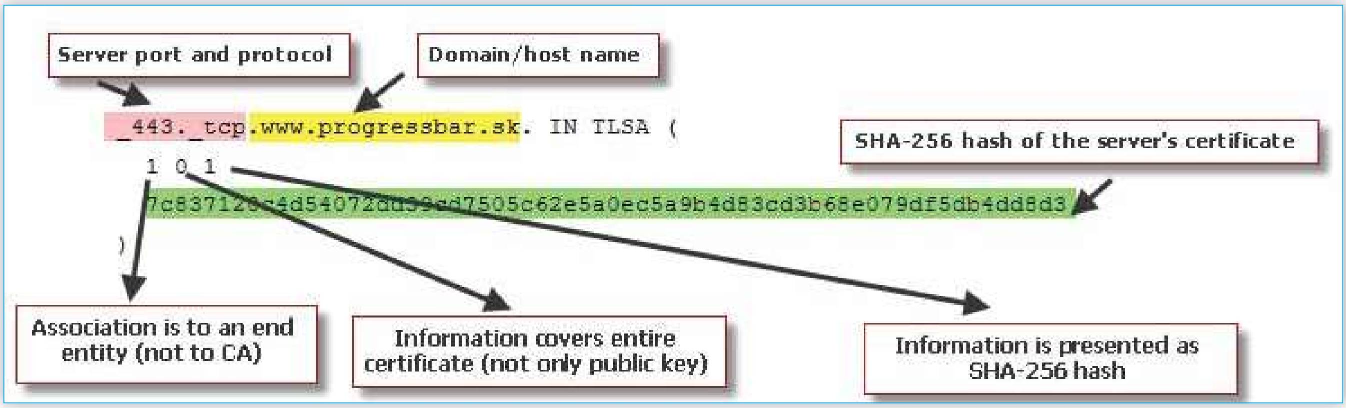
Conclusion

The main purpose of this article was to describe the basics of SSL/ TLS usage, its problems with regards to cryptographic key management and outline possible solutions. There is no easy way to solve the problem of cryptographic key management and we will see which solution will be accepted by the industry and internet users.

Besides the existing solution proposals new ones are emerging, especially EFF Sovereign Keys and Ben Laurie's and Adam Langley's work on Auditable CAs are worth mentioning. We shall follow the situation and bring you more information on the topic. In the mean time we have informed you about existing detection mechanisms (Certificate Patrol, Perspectives, Convergence) so that you can browse the internet in a safer way.

MARTIN RUBLÍK, PH.D., CISSP

Martin Rublík is a senior security consultant at BSP Consulting and a lecturer at University of Economics in Bratislava. He received his master's degree at Faculty of Mathematics, Physics and Informatics (Comenius University in Bratislava) in 2005 and his Ph.D. degree at Faculty of Economic Informatics (University of Economics in Bratislava) in 2010. His main area of expertise includes identity management, PKI, smart cards and network security.



```
_443._tcp.www.progressbar.sk. IN TLSA (
    1 0 1
    7c837128c4d54072dd59cd7505c62e5a0ec5a9b4d83cd3b68e079df5db4dd8d3
    )
```

Server port and protocol
Domain/host name
SHA-256 hash of the server's certificate
Association is to an end entity (not to CA)
Information covers entire certificate (not only public key)
Information is presented as SHA-256 hash

Figure 3. *Sample DANE TLSA RR*