Efficient Handling of Certificates within Public-Key Infrastructures

E. FALDELLA, M. PRANDINI
DEIS – Dept. of Electronics, Computer Sciences and Systems
University of Bologna
Viale Risorgimento, 2 – 40136 Bologna
ITALY

Abstract: - The paper addresses the problem of efficiently handling certificates within public-key infrastructures, from both the communication traffic and computational load points of view. The main state-of-the-art schemes, recently proposed from both the academic and the industrial world, are discussed and the most relevant security, timeliness and efficiency features outlined. A thorough analytical investigation on the attainable performances is then carried out. Particular emphasis is placed on the computational load deriving from application of each scheme, because this matter has not yet received the attention it deserves.

Key-Words: - Legally valid documents, digital signatures, X.509 certificates publication and revocation, Public-Key Infrastructures, performance evaluation. CSCC'99 Proceedings, Pages:7331-7338

1 Introduction

In the last few years the Internet has become a ubiquitous reality, widely used to carry all kinds of public-domain information. With the integration of adequate security features in the existing infrastructure, its diffusion and ease of access could be exploited to effectively enhance also the exchange of sensitive data, like legal documents and business transactions. Asymmetric cryptography, with its capability to provide privacy, authentication, integrity and non repudiation features for digital documents, seems to be the most viable solution presently available.

Within an asymmetric cryptosystem each user owns a pair of numbers called keys. Due to the mathematical properties of the pair, a digital signature produced with one key can be verified with the other one only. The key used for signing is called private key and must be kept strictly secret by its owner, while the key used to verify signatures is called public key and should be made widely available.

Public Key Infrastructures (PKIs) perform the fundamental task of binding each public key to its owner. The entities of a PKI and their mutual relationships, described in detail in [1], are schematically shown in Fig. 1. When a user presents a request to the Registration Authority (RA) to become a PKI member, a certificate is issued which contains the user identity (ascertained in a secure way) and the associated public key, together with some other relevant information such as the certificate issue and expiration dates. The certificate is signed by a Certification Authority (CA), so that its integrity is guaranteed, and published to the directory where it is made available to all PKI users. The CA's public key, which is required to verify any CA's signed document, is transferred to the new user via a secure channel.

A study [2] undertaken on behalf of the USA National Institute for Standard and Technology (NIST) shows that, on the average, 10% of all certificates managed within a PKI are subject to revocation, i.e. they lose validity before the natural expiration date (commonly set to one year after the emission date). Revocation can be necessary for many reasons, e.g., private key compromise or changes in user affiliation. As a consequence, proper use of a public-key certificate involves not only integrity, authentication and expiration date verification, but also revocation status check.

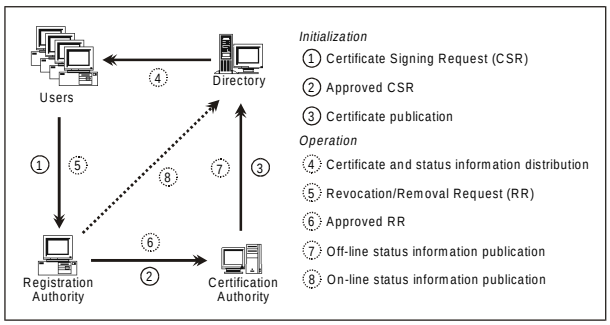

Fig. 1 - Architectural model of a PKI

A very important classification is between off-line and on-line certificate status handling systems. The definition of off-line systems encompasses all the schemes where a CA, not exposed to network attacks, periodically (typically once per day) authenticates and publishes to the directory the certificate status changes notified by the RA. The main advantage of these schemes is that they don't rely on a trusted directory, the integrity of status information being directly guaranteed by the CA's signature. The main drawback, on the other side, is the latency associated with status information updating following either a certificate revocation or a certificate revocation removal. On-line systems overcome this limitation by temporarily delegating the task of authenticating certificate status information to the directory. Within these schemes, in fact, status updating is performed by the directory not only periodically on the basis of the information received by a CA, but also on-line, following direct notification by the RA of a certificate status change. The drawback now is that the directory must be trusted and suitably protected against network attacks.

Whichever scheme is chosen, the deployment of a certificate status handling system deeply affects the performance of a PKI. This clearly emerges from the NIST study cited above, which indicates that the communication traffic related to certificate status updating and verification is the cause of the highest among the PKI maintenance costs. Other parameters, such as the computational load deriving on PKI's entities from application of a certificate status handling scheme, greatly influence overall performance. In the following the main recently proposed schemes from both the academic and the industrial world, namely the Certificate Revocation List [1], the Certificate Revocation Status [3], the Certificate Revocation Tree [4], and the On-line Certificate Status Protocol [5], are examined. For each of them the most relevant security, timeliness and efficiency features are outlined, the latter being emphasized in terms of:

- directory incoming and outgoing traffic;
- CA, directory (when applicable) and user computational load.

Comparative analysis of performance has been performed under the NIST assumptions that the total number of certificates handled within a PKI is constant and uniformly distributed among CAs, and that each CA works under stationary conditions as regards the number of certificates subject to revocation. This means that, for each CA and within any certificate status update period, the average number of new revocations is balanced by as many revocation removals. As regards PKI's entities computational load, a matter often not explicitly addressed in the literature, a thorough investigation has been purposely carried out. In order to make the results more coherent and consequently their comparison more meaningful, a common metric has been adopted for characterizing the load associated with the different cryptographic primitives (such as hash functions, lightweight and standard signatures) exploited by the various schemes.

2 Off-line certificate status handling schemes

2.1 Certificate Revocation List (CRL)

CRL, currently published as RFC2459 [1] in the Internet Society standards track, has been the first proposed scheme for certificate status handling. Within this scheme every CA maintains a list of (the serial numbers of) all assigned revoked certificates, each associated with the corresponding revocation date and time. At a predefined daily rate *T*, the whole list, prefixed by the CA identity, the current date and time and the next scheduled certificate status publication date and time, is digitally signed and transferred to the directory. This very simple scheme clearly guarantees against any possible tampering of certificate status information. However it exhibits a serious drawback, since each time users query the directory, usually to know the status of a single certificate only, get a whole list as a reply.

(a) CA computational load evaluation

Under the assumption that each CA works under stationary conditions as regards the number of certificates subject to revocation, it is immediate to evaluate the entries number *R* of the list a CA has to periodically sign and transfer to the directory. *R* is simply given by:

$$R = \frac{N \cdot P}{N_{CA}} \tag{1}$$

where, according to NIST notation, *N* indicates the total number of certificates handled within a PKI, *P* the revoked certificates fraction, and N_{CA} the number of CAs. Each CRL entry involves l_{sn} bits for the representation of the serial number of a revoked certificate, plus l_{ts} bits for the representation of the

corresponding timestamp. A CA, whenever signs the list, has to first compute the list digest via a hash function, then perform a modular exponentiation on the result. By ignoring the contribution deriving from the information prefixed to the list, the CA daily computational load can thus be estimated as:

$$L_{CA} = T\left((l_{sn} + l_{ts}) \cdot R \cdot L_h + L_{sig}\right) \quad (2)$$

where L_h denotes the equivalent load of a 1-bit digest operation and L_{sig} the equivalent load of a modular exponentiation.

(b) Directory incoming traffic evaluation
At each status update, every CA publishes its list on the directory. The deriving directory incoming traffic, in bit/day, is then given by:

$$T_{DIR-IN} = T \cdot \left((l_{sn} + l_{ts}) \cdot R + l_{sig}\right) \cdot N_{CA} \quad (3)$$

where l_{sig} indicates the number of bits involved in the representation of a signature.

(c) Directory computational load evaluation
In a CRL-based system no cryptographic operations are requested to the directory in order to reply to user queries.

(d) Directory outgoing traffic evaluation
Each time a user needs to check a certificate status, a whole list is sent as reply. The overall directory outgoing traffic, in bit/day, is then given by:

$$T_{DIR-OUT} = Q \cdot \left((l_{sn} + l_{ts}) \cdot R + l_{sig}\right) \quad (4)$$

where Q is the daily number of user queries.

(e) User computational load evaluation
To perform a certificate status check, a user needs to search the CRL for the corresponding serial number, once verified the signature on the list. It is reasonable to assume the latter contribution as the most relevant, so that the computational load can be estimated as:

$$L_{USER} = (l_{sn} + l_{ts}) \cdot R \cdot L_h + L_{sig} \quad (5)$$

2.2 Certificate Revocation Status (CRS)

With the aim of preventing users being flooded with superfluous information, Micali proposed CRS in 1995. This scheme exploits as cryptographic primitive the lightweight signature introduced by Lamport in 1981 [6], which allows to sign a limited set of elements more efficiently than standard general-purpose signatures. A lightweight signature algorithm is exploited by CAs to individually sign, at each updating period, each single certificate status. This operation, which would bear an unsustainable load if performed with standard signatures, allows replying to user queries only with the exact piece of information needed.
In details, for each certificate a CA generates and keeps strictly secret a pair of numbers YES_0 and NO_0. A public domain one-way hash function f is applied both to NO_0 to get $NO=f(NO_0)$, and, repeatedly, to YES_0 to get $YES=f^n(YES_0)$, where n is the number of status updates envisaged during the certificate lifetime. *NO* and *YES* are published as part of the certificate. At the i^{th} status updating period, a CA computes and publishes either $S_i=f^{n-i}(YES_0)$ or $S_i=NO_0$, depending on the certificate being still valid or revoked, respectively. By getting S_i as a reply to a status query, a user is able to determine the validity of a certificate simply checking whether $f(S_i)=NO$ or $f^i(S_i)=YES$ (Fig. 2). Phony extension of the validity of a revoked certificate (or phony revocation of a valid certificate) without knowledge of YES_0 (NO_0) is impossible, as it would imply inversion of the one-way function. It is clear that this scheme can be used n times only, and that status updating timing must be strictly fixed to avoid replay of old replies.
The advantage of this solution over CRL is the strong reduction in communication traffic between directory and users. However, notwithstanding the use of lightweight signatures, the CA computational load is much higher than in CRL. The deriving directory incoming traffic is also much higher, since it is no more proportional to the number of revoked certificates, but to the total number of certificates handled within a PKI.

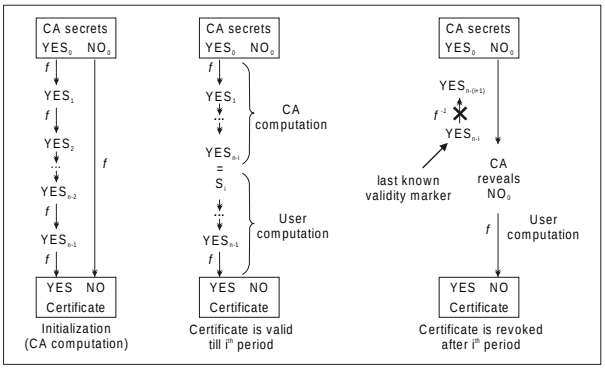

Fig. 2. CRS scheme operations.

(a) CA computational load evaluation
In the usual case of one-year certificate lifetime, every CA has to periodically apply the one-way function, on the average, 365·*T*/2 times for each of the (1-*P*)·(*R*/*P*) valid certificates. By ignoring the additional load due to revoked certificates and the initial computation of *YES* for the new certificates, the CA daily computational load can thus be estimated as:

$$L_{CA} = \frac{365}{2} \cdot T^2 \cdot \frac{1-P}{P} R \cdot l_{lwsig} \cdot L_h \qquad (6)$$

where l_{lwsig} is the number of bits involved in the representation of the validity status of a certificate.

(b) Directory incoming traffic evaluation
At each status update, the directory receives a pair (serial number, updated *YES* value) for each valid certificate, and a pair (serial number, NO_0 value) for the each certificate just revoked. The resulting traffic, in bit/day, is therefore:

$$T_{DIR-IN} = T \cdot (l_{sn} + l_{lwsig}) \cdot \left(N \cdot (1-P) + \frac{N \cdot P}{365 \cdot T} \right) \qquad (7)$$

(c) Directory computational load evaluation
In the CRS scheme no cryptographic operations are requested to the directory in order to reply to user queries.

(d) Directory outgoing traffic evaluation
A user query about a certificate is replied with the corresponding status information only. The resulting directory outgoing traffic, in bit/day, is therefore:

$$T_{DIR-OUT} = Q \cdot l_{lwsig} \qquad (8)$$

(e) User computational load evaluation
A user is asked to perform the computation needed to transform the returned status information either into *NO* or into *YES*. The first case, which requires a single application of the hash function, occurs with probability *P*; the second case, which requires on the average 365·*T*/2 applications of the hash function, occurs with probability (1-*P*). The resulting average computational load is therefore:

$$L_{USER} = \left(P + \frac{365T}{2}(1-P) \right) \cdot l_{lwsig} \cdot L_h \qquad (9)$$

2.3 Certificate Revocation Tree (CRT)

The more recent CRT method, devised by Kocher, exploits a new concept. In this scheme the status of the certificates handled by a CA is represented by partitioning the domain of the associated serial numbers into as many subranges as the number of revoked certificates. Each subrange is represented by a statement simply reporting its bounds, meaning that only the certificate at the lower bound is revoked. A single statement thus provides an explicit status proof for all certificates belonging to the related subrange.
A statement set is authenticated by building a binary tree, where the leaves are associated with the statements and each intermediate node is computed as the cryptographic hash of the concatenation of the corresponding two sons (Fig. 3). The tree root, which eventually contains a contribution from all the statements, is signed. The status information about a certificate is derived from the leaf node associated with the statement encompassing the certificate serial number, the corresponding sibling leaf node and all other siblings of the nodes in the search path leading to the tree root. Due to the one-way property of the hash function, it is not possible to forge a statement and find a corresponding proof of existence leading to the authenticated root value.
Certificate status update forces a CA to rebuild the whole tree, with complexity proportional to the number of revoked certificates. By exploiting the capability of the directory to do the same, it is possible to drastically reduce the traffic from CAs to directory, by transferring only the serial number of the certificates whose status has changed since the last update.

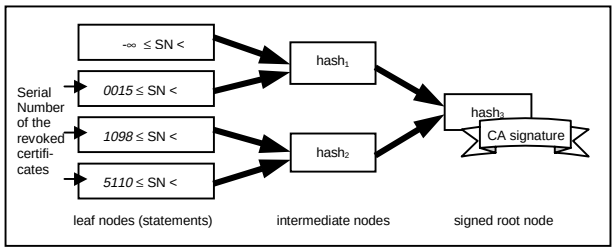

Fig. 3. Example of a 4-leaves CRT.

(a) CA computational load evaluation
At each status update, a CA has to rebuild a binary tree with as many leaves as the number *R* of revoked certificates. This calls for computation of (*R*-1) hashes, each compressing two nodes of size l_{stat} into one, followed by a digital signature operation

performed over the tree root. The resulting daily computational load is therefore:

$$L_{CA} = T \cdot \left(2\,(R-1) \cdot l_{stat} \cdot L_h + L_{sig}\right) \quad (10)$$

(b) Directory incoming traffic evaluation
Two alternative approaches are possible for updating certificate status information on the directory. If the directory computational load is to be minimized, or if the directory has no computation capabilities at all, the whole tree, together with the signed root, has to be published by each CA. If the capability of the directory to build a tree can be exploited, each CA has to send, in addition to the tree signed root, only the serial number of certificates whose status changed since the last update. Since communication traffic is usually the most critical factor, the second approach is here examined. The directory incoming traffic, in bit/day, is given by:

$$T_{DIR-IN} = T \cdot (\frac{2}{365T} R \cdot l_{sn} + l_{sig}) \cdot N_{CA} \quad (11)$$

(c) Directory computational load evaluation
The directory has to build, at each update, N_{CA} trees, leading to a daily computational load of:

$$L_{DIR} = T \cdot 2\,(R-1) \cdot l_{stat} \cdot L_h \cdot N_{CA} \quad (12)$$

(d) Directory outgoing traffic evaluation
The reply to a user query about a certificate status comprises the related statement, its proof of existence (composed of as many nodes as the tree depth), and the signature on the tree root. The directory outgoing traffic, in bit/day, is given by:

$$T_{DIR-OUT} = Q \cdot \left((1 + \lceil \log_2 R \rceil) \cdot l_{stat} + l_{sig}\right) \quad (13)$$

(e) User computational load evaluation
A user, receiving the reply described above, computes the proper sequence of hash operations and verifies the signature over the resulting value. The computational load is:

$$L_{USER} = 2 \cdot \lceil \log_2 R \rceil \cdot l_{stat} \cdot L_h + L_{sig} \quad (14)$$

2.4 Certificate Revocation Tree Extension

An evolution of CRT, based on a data structure called 2-3 tree, has been recently proposed by Naor and Nissim [7]. The advantage of this structure over a binary tree is that node insertion or deletion can be performed without rebuilding the whole tree: only the search path leading to the new or old node is affected. The computational load induced by status updating is consequently lower, whereas the directory outgoing traffic is higher because in 2-3 trees a node may have either one or two sibling nodes. Moreover, in the Naor-Nissim scheme the tree leaves represent revoked certificates instead of statements. Therefore the proof of a certificate validity calls for the demonstration of the existence of two adjacent leaves, representing respectively a revoked certificate with serial number lower and one with serial number higher than the queried one.

3 On-line certificate status handling schemes

3.1 On-line Certificate Status Protocol (OCSP)

OCSP derives from the original proposal of the Real-Time Certificate Status Protocol [8]. Within an OCSP-based system certificate status authentication is delegated to a responder within the directory, i.e. it is guaranteed by a signature produced with a key that a CA and/or users trust. The certificate status database can be periodically updated by means of traditional CRL issuing, or also by means of immediate notification of a status change request from the Registration Authority. This strategy not only improves information updating timeliness, but also reduces the directory outgoing traffic, because a reply contains only status information about the queried certificate.

(a) CA computational load evaluation
CAs are not directly involved in the generation of OCSP replies.

(b) Directory incoming traffic evaluation
The OCSP responder needs to receive revocation and revocation removal notices only. The incoming traffic, in bit/day, is given by:

$$T_{DIR-IN} = \frac{2N \cdot P \cdot l_{sn}}{365} \quad (15)$$

(c) Directory computational load evaluation
Each query needs a signed reply which, the first time, has to be computed on the fly. Once computed, replies can be cached and replayed for some time. This advantage, however, can be ignored when,

according to the NIST working assumptions, a certificate is queried about once a day. A one-day-old reply will probably be considered too old, and will be signed again when needed. The daily computational load is then:

$$L_{DIR} = Q \cdot (l_{OCSP\,r} \cdot L_h + L_{sig}) \quad (16)$$

where l_{OCSPr} is the number of bits needed to represent an OCSP response, formed with the queried certificate serial number, the status indication (good, revoked or unknown), and various header information.

(d) Directory outgoing traffic evaluation
The directory outgoing traffic derives from the transmission of the signed replies and is given, in bit/day, by:

$$T_{DIR-OUT} = Q \cdot (l_{OCSP\,r} + l_{sig}) \quad (17)$$

(e) User computational load evaluation
A user needs only to check the reply signature:

$$L_{USER} = l_{OCSP\,r} \cdot L_h + L_{sig} \quad (18)$$

3.2 Proprietary certificate extensions

Various PKI-related software developers have defined their own implementations of certificate status handling schemes. An example is the Netscape Certificate Extension [9], which exploits the capability of X.509v3 certificates to contain optional data fields called extensions. Including a *netscape-revocation-url* extension in a certificate causes any Netscape software to contact the specified location in accordance with the HTTP protocol and perform a status query. (The extension is simply ignored when the certificate is parsed by non-Netscape software.) The reply should consist of a single ASCII character, a '0' if the certificate is currently valid, a '1' in the opposite case. The security of this scheme is not very high, because the reply is not signed at all. The only kind of guarantee about its integrity and authenticity comes from the use of a secure (HTTPS) connection.

4 Concluding remarks

The performances of the considered state-of-the-art certificate status handling schemes are synthetically illustrated in Figs. 4-7. The numeric values selected for quantitative comparison (see Table 1) have been derived from NIST suggestions and from publicly available benchmarks on cryptographic algorithm execution times [10]. The on-line OCSP scheme provides the highest timeliness, at the expense, however, of a heavily loaded directory, and exhibits a fair level of security when supported by parallel emission of CA-authenticated certificate status information, like CRLs. More recently proposed off-line schemes yield very interesting performance. They are intrinsically more secure than any on-line scheme and exhibit, particularly CRT, a communication traffic not much higher than OCSP and a computational load low enough to allow frequent updating of certificate status information.
Research studies currently being undertaken aim to devise off-line schemes that make both the communication traffic, particularly the directory incoming traffic, and the overall computational load less dependent from the number of PKI users and certificates status update frequency. Interesting results seem emerge from approaches exploiting OWA cryptographic primitives [11], [12] and incremental cryptography techniques [13-16].

Symbol	Meaning	Default value
N	total number of certificates	$3 \cdot 10^6$
P	revoked certificates fraction	0.1
N_{CA}	number of CAs	100
T	daily number of certificate status updates	1
Q	daily number of certificate status queries	= N
l_{sn}	certificate serial number bits number	20
l_{ts}	timestamp bits number	48
l_{sig}	digital signature bits number	1024
l_{lwsig}	CRS status information bits number	100
l_{stat}	CRT statement bits number	128
$l_{OCSP\,r}$	OCSP response bits number	100
L_h	MD5 hash computation time (inverse of bitrate)	2.2 ns
L_{sig}	RSA-1024 signature computation time	27 ms

Table 1 - Parameter values used for comparative analysis of performance

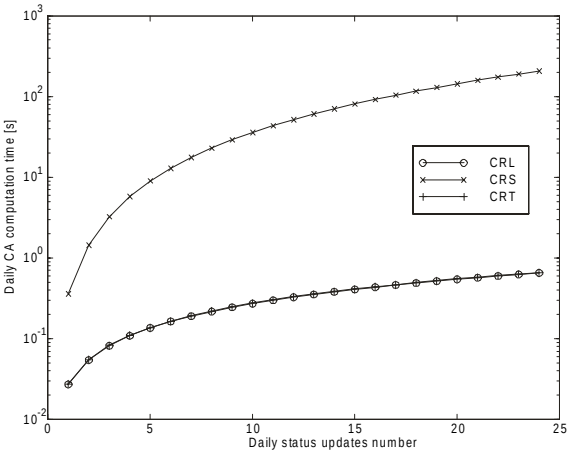

Fig. 4a. CA computational load dependency on certificate status update frequency .

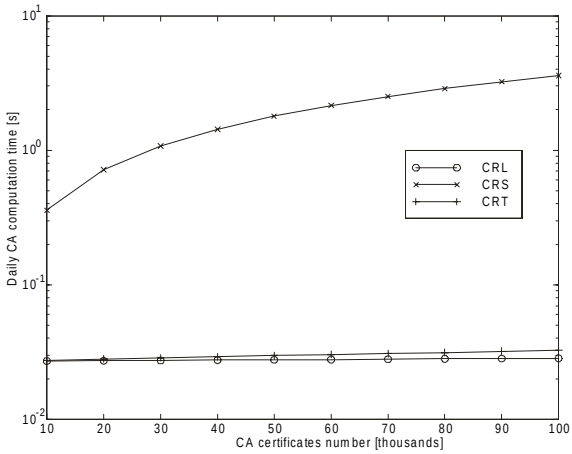

Fig. 4b. CA computational load dependency on certificates number.

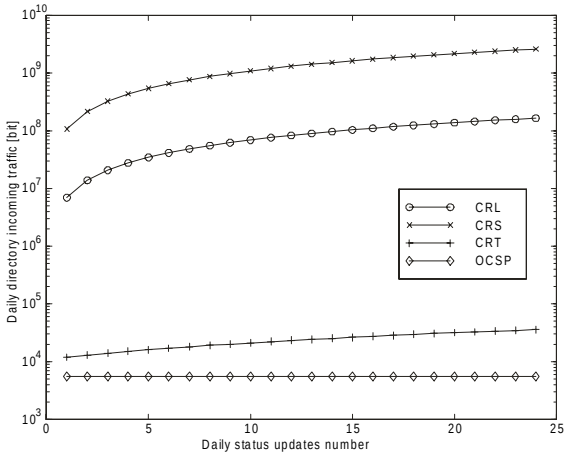

Fig. 5a. Directory incoming traffic dependency on certificate status update frequency.

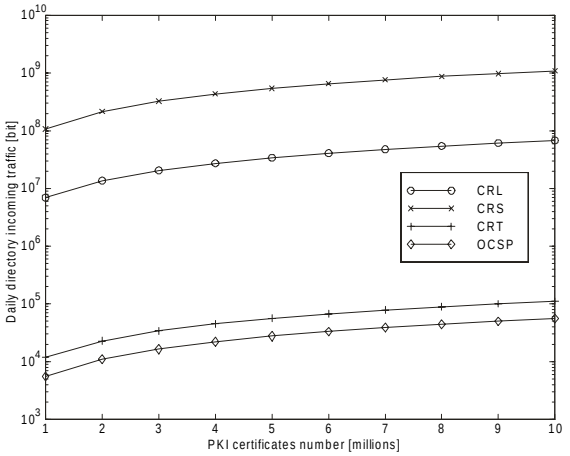

Fig. 5b. Directory incoming traffic dependency on certificates number

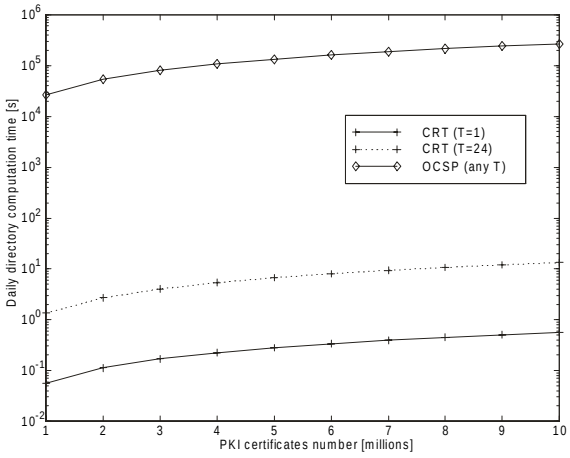

Fig. 6a. Influence of certificates number on directory computational load.

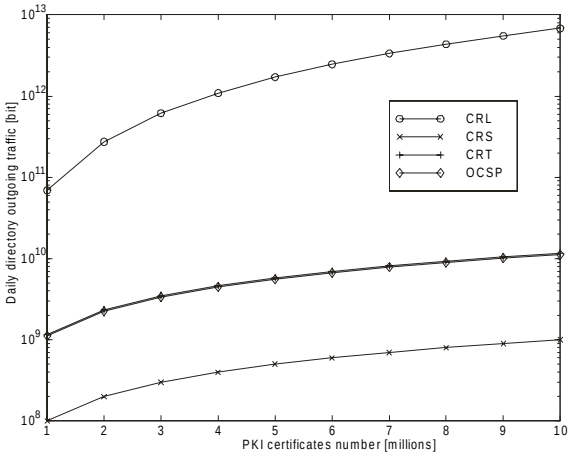

Fig. 6b. Influence of certificates number on directory outgoing traffic.

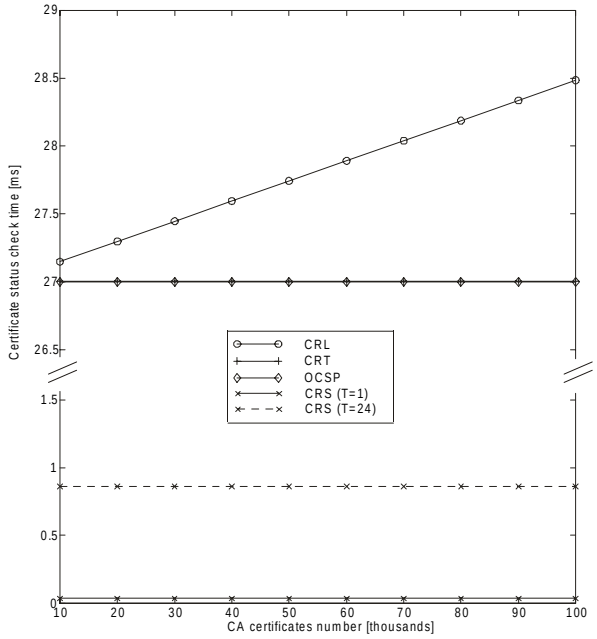

Fig. 7. Influence of certificates number on user computational load.

References

[1] R. Housley, W. Ford, W. Polk, D. Solo, RFC 2459 – Internet X.509 Public Key Infrastructure Certificate and CRL Profile, The Internet Society, Jan. 1999. (http://www.rfc-editor.org/rfc/rfc2459.txt)

[2] Public-key Infrastructure Study, NIST, Gaithersburg, MD, April 1994.

[3] S. Micali, Efficient Certificate Revocation, Technical Memo MIT/LCS/TM-542b, 1996.

[4] P. Kocher, A Quick Introduction to Certificate Revocation Trees (CRTs). (http://www.valicert.com/ resources/bodyIntroRevocation.html)

[5] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, IETF, Sept. 1998. (ftp://ftp.ietf.org/internet-drafts/draft-ietf-pkix-ocsp-07.txt)

[6] L. Lamport, Password Authentication with Insecure Communication, *Communications of the ACM*, v. 24, n. 11, Nov. 1981, pp. 770-772.

[7] M. Naor, K. Nissim, Certificate Revocation and Certificate Update, *Proceedings 7th USENIX Security Symposium*, 1998.

[8] A. Malpani, C. Adams, R. Ankney, S. Galperin, Internet Public Key Infrastructure Real Time Certificate Status Protocol – RCSP, IETF, Mar. 1998. (ftp://ftp.ietf.org/internet-drafts/draft-malpani-rcsp-00.txt)

[9] J. Weinstein, Netscape Certificate Extensions Specification Draft. (http://form.netscape.com/eng/security/comm4-cert-exts.html)

[10] W. Dai: Speed Comparison of Popular Crypto Algorithms (http://www.eskimo.com/~weidai/ benchmarks.html

[11] J. Benaloh, M. de Mare, One-Way Accumulators: A Decentralized Alternative to Digital Signatures (Extended Abstract), *Eurocrypt '93 Proceedings*, pp. 274-285.

[12] E. Faldella, M. Prandini, An Extension of the One-Way Accumulator Cryptographic Primitive for Efficient Handling of Certificates in PKIs – submitted to *ACM Transactions on Information and System Security*.

[13] M. Bellare, O. Goldreich, S. Goldwasser, Incremental Cryptography: the Case of Hashing and Signing, *Crypto '94 Proceedings*, Lecture Notes in Computer Science, v. 839, pp. 216-233, Springer-Verlag, 1994.

[14] M. Bellare, O. Goldreich, S. Goldwasser, Incremental Cryptography and Application to Virus Protection, *Proceedings of the 27th Annual ACM Symposium on the Theory of Computing*, pp. 45-56, 1995.

[15] M. Bellare, R. Guerin, P. Rogaway, XOR MACs: New Methods for Message Authentication Using Finite Pseudorandom Functions, *Crypto '95 Proceedings*, Lecture Notes in Computer Science, v. 963, pp. 15-29, Springer-Verlag, 1995.

[16] M. Fischlin, Incremental Cryptography and Memory Checkers, *Eurocrypt '97 Proceedings*, Lecture Notes in Computer Science, Vol.1233, pp.393-408, Springer-Verlag, 1997.