A Novel Approach to On-Line Status Authentication of Public-Key Certificates

Eugenio Faldella Marco Prandini
Department of Electronics, Computer Science and Systems
University of Bologna
Bologna, Italy
{efaldella, mprandini}@deis.unibo.it

Abstract

The widespread use of public networks, such as the Internet, for the exchange of sensitive data, like legally valid documents and business transactions, poses severe security constraints. The approach relying on public-key certificates certainly represents a valuable solution from the viewpoint of data integrity and authentication. The effectiveness of the approach, however, may be arguable, especially when a trivial strategy is adopted within a Public-Key Infrastructure (PKI) to deal with the problem of revoked certificates.

This paper presents a novel certificate status handling scheme, based on a purposely-conceived extension of the One-Way Accumulator (OWA) cryptographic primitive. The distinguishing characteristic of the devised Owa-based Revocation Scheme (ORS) is that it exploits a single directory-signed proof to collectively authenticate the status of all the certificates handled by a Certification Authority (CA) within a PKI. A thorough investigation on the performance attainable shows that ORS exhibits the same features of the well-known On-line Certificate Status Protocol (OCSP) as regards security, scalability and certificate status-updating timeliness, at the same time drastically reducing the directory computational load that, in a high-traffic context, could be nearly unbearable when OCSP is applied.

1. Introduction

The support for secure data exchange over public networks is the prerequisite for a wide range of important applications, like e-commerce. PKIs have been proposed as a scalable solution to the problem of securely connecting two unrelated entities, allowing them to trust each other's identity. This goal is attained by means of public-key certificates [1], i.e. digital documents that bind the identification data of an entity with its public key. The identification data is securely ascertained by a Registration Authority (RA), which also verifies that the entity asking for certification of a public key holds the corresponding private key. A CA digitally signs the certificate to make the binding effective for a given period of time, usually one year. As the last issuing step, the certificate is made available to all PKI users via publication on a public directory.

A certificate may be invalidated before its foreseen expiration date due to many possible events, like changes in the owner identification data (e.g., affiliation) or private key compromise. Proper usage of a certificate consequently involves not only its formal verification, but also revocation status checking. Of course, certificate status updating and verification induce computational load over the PKI entities and network traffic between them. According to NIST estimates [2], the deriving costs could be in excess of the 90% of the overall PKI costs when a trivial revocation scheme is adopted.

To deal with this outstanding problem, a number of revocation schemes have been recently proposed [3,4]. They can be classified as either off-line or on-line schemes, depending on which PKI entity is deputed to certificate status authentication: a CA, usually disconnected from the network, in the former schemes; the directory, permanently connected to the network, in the latter. The main positive feature of off-line schemes, such as CRL [1], CRS [5] and CRT [6], is the capability of producing CA-guaranteed certificate status information which can be forged by neither external attackers, nor a malicious or compromised directory. The drawback is the lack of certificate status-updating timeliness, due to the high network traffic and computational load deriving from their application. For this reason on-line schemes, such as OCSP [7], have recently received an increasing attention. These schemes are capable of reflecting any certificate status change in real-time by delegating the task of certificate status authentication directly to the directory.

This paper presents a novel on-line scheme based on a purposely-conceived extension of the OWA cryptographic primitive, which exhibits the same performances of OCSP as regards security, scalability and certificate status-updating timeliness. The distinguishing characteristic of the devised scheme is that it exploits a single directory-signed proof to collectively authenticate the status of all the certificates handled by a CA within a PKI. As a consequence, the directory computational load, that in a high-traffic context could be unbearable when OCSP is applied, is drastically reduced.

In the next sections the OWA cryptographic primitive is first recalled and its extension described, then the new ORS certificate revocation scheme is introduced. A detailed comparative analysis of the performance attainable with ORS and OCSP is finally illustrated.

2. The One-Way Accumulator cryptographic primitive and its extension

The notion of OWA, introduced by Benaloh and de Mare [8], is, briefly, that of a function h_n, characterized by a security parameter n and computable in a polynomial time in n, which is *quasi-commutative*:

$$h_n : X_n \times Y_n \rightarrow X_n$$
$$h_n(h_n(x, y), y') = h_n(h_n(x, y'), y) \quad \forall\, x \in X_n, \forall\, y, y' \in Y_n.$$

By means of OWA it is possible to define a membership test with a structure similar to that proposed by Diffie and Hellman for group key agreement [9,10]. Let $y_1, y_2, \ldots, y_m$ be the identifiers of the m members of a group, and x a parameter the members agreed upon. Once exchanged the identifiers, each member computes

$$z_i = h_n(h_n(\ldots(h_n(x, y_1), y_2), \ldots), y_m)$$

without its own y_i. Let us call z_i *complement* of y_i. Since the function h_n is quasi-commutative, $h_n(z_i, y_i)$ will hold the same value (i.e., the OWA computed over all y_i) for each member i. To prove its membership, a member i presents its own pair (y_i, z_i) to another member j, who can immediately verify that $h_n(z_i, y_i)$ equals $h_n(z_j, y_j)$. The advantage of this technique is that remembering the whole members' list is not required. Furthermore, the OWA could be revealed to third parties, enabling them to recognize a group member without knowing the members' list.

The suitability of OWA as a cryptographic primitive is founded on the difficulty of inverting h_n, i.e., given a pair (x, y) chosen uniformly from $X_n \times Y_n$ and any y' chosen uniformly from Y_n, it should not be possible to find with probability higher than $1/P(n)$, for every polynomial P, an $x' \in X_n$ such that $h_n(x, y) = h_n(x', y')$.

The function identified by OWA authors is:

$$h_n(x, y) = x^y \bmod n,$$

which satisfies the above mentioned requirements if n is appropriately chosen, as shown in [11] for RSA. When the modular exponentiation is repeatedly applied, it is necessary to select n in a more restrictive way in order to avoid that the image size is reduced to such an extent that finding collisions becomes feasible. For this purpose n should belong to the *rigid integers* set, defined by:

"rigid integers" :=
{ $n \mid n = p\,q$, *p and q are distinct "safe integers"* },
"safe integers" :=
{ $s \mid$ *s is prime,* $s = 2s' + 1$, *s' is an odd prime* }.

Not considering the procedural aspects for the admission, aggregation of a new member k to the group requires the distribution of its identifier y_k, which has to be accumulated by each old member on the corresponding complement to derive the new value (marked with the prime) that continues to satisfy the membership test:

$$z_i' = h_n(z_i, y_k) \qquad \forall\, i.$$
$$h_n(z_i', y_i) = OWA'.$$

To remove a member k from an OWA, a set of operations must be performed on the OWA and on the complement of the other members in such a way to get a new set of values (marked with the prime) such that:

$$h_n(z_i', y_i) = OWA' \qquad \forall\, i \neq k,$$
$$h_n(z_k, y_k) \neq OWA'.$$

That is, each member left continues to satisfy the membership test, while the removed member does not. Furthermore, the knowledge of z_k, y_k and OWA' should not enable to compute a value z_k' such that $h_n(z_k', y_k) = OWA'$. OWA authors do not consider the problem of member removal, and Schneier [12] suggests that it could simply be ignored. Actually, in a totally decentralized scenario, this problem appears to only have an expensive solution, consisting in repeating the initial step of information exchange, so that each member can once again compute its complement. It is not too difficult to update the complements every time a member is removed when, conversely to the original application, the members' list can be kept in a central site. This condition comes upon in a PKI, where computing centers are responsible for certificate status management.

If the list is centrally kept, all the complements can be computed again over the new set, so that, after a removal,

all the security conditions are guaranteed to hold by the same assumptions on which OWA is based (the contribution of y_k is simply lost). This computation can be effectively enhanced under the assumption that the center performs OWA initialization, i.e., that it selects x and (the prime factors of) n. Instead of rebuilding all complements from scratch, it is possible to *neutralize* the contribution of y_k on each z_i ($i \neq k$). In fact, due to the quasi-commutative property of the modular exponentiation, z_i can be thought of as $(z'_i)^{y_k} \bmod n$, and therefore z'_i may be directly computed from z_i by taking its y_k^{th} root mod n, or, with the same effect, by raising it to y_k^{-1}. This reverse computation is possible if and only if y_k is relatively prime to the Euler's totient function $\Phi(n)$, which in the case of a rigid integer $n = (2p'+1)(2q'+1)$ is given by $4p'q'$. This is not a tight constraint in our opinion, because the probability p_{np} that a number y_k uniformly chosen among the odd numbers smaller than $\Phi(n)$ has a common factor with $\Phi(n)$ can be arbitrarily reduced by increasing n:

$$p_{np} = \frac{\Phi(n)/2 - \Phi(\Phi(n))}{\Phi(n)/2} = 1 - \frac{2(p'-1)(q'-1)}{2p'q'} = \frac{p'+q'-1}{p'q'} \approx O(n^{-1/2}).$$

In particular, all the odd numbers smaller than both p' and q', being relatively prime to 4, p', and q', are relatively prime to $\Phi(n)$.

Some remarks can be highlighted regarding the complexity of the aforementioned operations. In both cases (list rebuilding and member neutralization), multiple removal of members does not increase the number of operations to be performed. In fact, if the list is rebuilt, the higher the number of removed members, the lower the number of identifiers to accumulate when evaluating the new complements. If neutralization is used, a single modular exponentiation is needed for updating each complement, once the product of the inverse identifiers to be removed is computed mod $\Phi(n)$. The latter technique can be efficiently exploited for simultaneous multiple insertion and removal of members. A single modular exponentiation is still required for each complement, the *cumulative exponent* being computed as the product, mod $\Phi(n)$, of the identifiers to be inserted and of the inverse identifiers to be removed.

3. Certificate status management in ORS

Our proposal for certificate status management is to consider the revoked certificates as members of an extended OWA. Like in the CRT scheme devised by Kocher [6] and in the conceptually similar Naor-Nissim scheme [13], the status of the certificates handled within 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 X_{low}, X_{high}, 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.

For authentication purposes, each statement is mapped to an identifier suitable to be accumulated to and possibly removed from an OWA. The identifier is obtained by concatenating a hash of the statement with a trailing 1. The resulting value is odd and its size is small enough (for instance 161 bits when using the SHA-1 hash function [14]) to guarantee that each possible identifier remains smaller than p' and q' for the most commonly adopted modulus sizes (at least 512 bits).

At any time the model representing the status of all certificates consists, for each CA handled with ORS, of:

- a set of statements spanning the serial number domain;
- the complement of each statement identifier;
- the OWA computed over all statement identifiers;
- the modulus n involved on its computation;
- a timestamp indicating the OWA production time;
- a signed digest of the last three items.

The model is updated according to requests notified to the directory by the RA. Issuing new certificates neither affects the statement set, nor leads to any change to the OWA and complements, since each usable serial number (i.e., not corresponding to a revoked certificate) already belongs to the scope of an existing statement. Changes to the statement set, and consequently to the complements and OWA, originate only from certificate revocation or revocation removal (for instance, due to its expiration). As regards updating of the statement set, it is necessary, in case of revocation, to replace the statement representing the subrange $[X_{low}, X_{high})$ containing the revoked certificate serial number X_r with the pair $[X_{low}, X_r)$ and $[X_r, X_{high})$, in case of revocation removal to perform the opposite substitution (Fig. 1).

As regards updating of the complements of unchanged statements, a modular exponentiation is required for each of them. This potentially huge computational load can be drastically reduced by maintaining, for each complement $z_i = x^{e_i} \bmod n$, an internal explicit representation of the two components x, e_i. In this way it is possible to reduce the overall number of exponentiations by updating only the exponent (via a modular multiplication by the cumulative exponent) and delaying the computation of a complement until it is actually needed in reply to a user

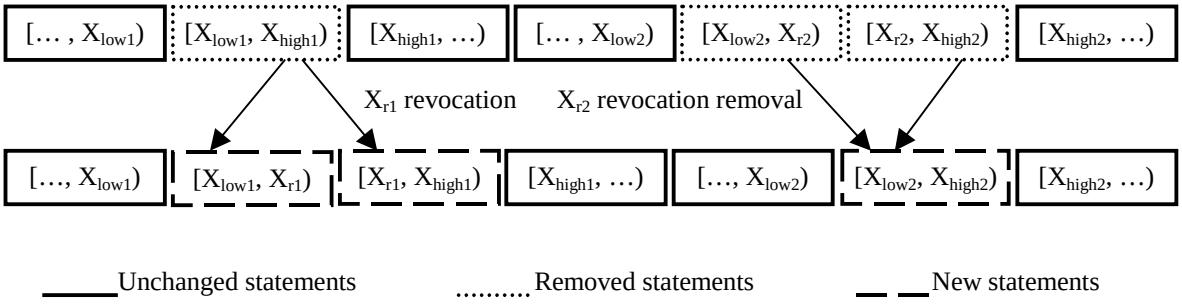

Fig. 1 – Statement set updating due to certificate revocation and revocation removal.

query. More important, it is possible to speed-up the exponentiation by exploiting its property of having a constant base *x*. The same procedure can be applied to the OWA, thus enabling very simple generation of the exponent for new statements, each being computed as the modular product of the OWA exponent by the statement identifier's inverse.

Regarding the interaction between the directory and users, a certificate status check is requested by indicating to the directory the selected serial number and the corresponding CA, and it is replied with the related statement, its complement, the timestamp, and the digest signature (the modulus *n* should conveniently appear into the directory's certificate). The verification process is straightforward: the user can easily compute the identifier associated with the statement, then the OWA and the digest of the original message (i.e., the sequence of the OWA, the modulus *n* involved on its computation, and the timestamp). This digest can be verified against the received signature.

The reply has all the desired features, that is it carries up-to-date information, it is of small fixed size and it is not forgeable. To forge a reply an attacker should compute a pair (statement identifier, complement) leading to the same authenticated OWA. This problem is generally not too difficult; however, if the specific action of altering the reply for a given query is considered, the computation becomes constrained. The forged statement must in fact contain two serial numbers chosen accordingly to both the user selected serial number and the altered meaning intended for the forged reply. Being the identifier a cryptographic hash of the statement, a tentative value may be changed but not selected. In this way, as emphasized in the security analysis performed by OWA authors [8], a probability of forgery well below 10^{-30} can be achieved over 20 million accumulated identifiers using a 220 digit modulus. For the proposed application a greater 1024-bit modulus has been considered, even if the involved statement set has a smaller cardinality (about $10^3/10^4$).

4. Performance evaluation

In on-line revocation schemes 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 is updated following immediate notification of a status change coming from the RA. Consequently, no computational load is induced over the CA and the directory incoming traffic is limited to the minimum amount of essential information.

The critical performance factor for on-line schemes is the directory computational load. In the OCSP scheme, the reply to each status query is signed together with a fresh timestamp. Caching is possible, but the benefits are unnoticeable, because the average time lapse between two queries for the same certificate is so long (one day according to NIST estimates) that a cached reply would contain a stale timestamp. The central idea in this work is to exploit an authentication scheme which can prove the freshness of all the possible replies by timestamping and signing a single item for each CA, that is the OWA.

The computational load deriving from the application of the ORS scheme is estimated by differentiating three different contributes:

1) computations needed for updating the internal status representation model following notification of status changes by the RA;
2) computations needed for replying to user certificate-specific queries;
3) computations involved by the authentication process.

In the following analysis of the three contributions, the symbols and values found in the NIST study modeling the federal PKI are used whenever applicable. In particular *N* represents the number of certificates within the PKI, *P* the fraction of revoked ones, *Q* the daily number of status queries, and N_{CA} the number of CAs within the PKI. The

number of certificates is assumed to be stationary, with the average daily number of revocations (*NP*/365) balanced by as many revocation removals.

1) As previously stated, each time a new revocation or revocation removal is notified, the exponent only of each complement corresponding to an unchanged statement (nearly the totality) needs updating, calling for globally NP/N_{CA} modular products. The computational load contributed by the generation of the exponents corresponding to new statements is negligible. The daily computational load, expressed in modular products, is then given by:

$$L_{update} = 2\frac{NP}{365}\cdot\frac{NP}{N_{CA}}$$

2) When a reply involving an outdated complement z_i is actually needed, the modular exponential $x^{e_i} \bmod n$ is computed exploiting the aforementioned optimization.

The widely known and quite efficient square-and-multiply modular exponentiation algorithm performs an average of 1.5*k* modular products, being *k* the number of bits involved in the representation of the exponent. The greatest part (*k*) of these products are needed to square the base at each step. If the base *x* is fixed and known, the values of its powers can be precomputed, achieving a tradeoff between the additional memory space requirements and the algorithm speedup. A straightforward choice is to build a *k*-entry table, storing at position $i \in [0,k\text{-}1]$ the value x^{2^i} (Fig. 2a). The deriving algorithm complexity is thus reduced to an average of 0.5*k* modular products. A more general and efficient solution can be designed, by grouping the exponent bits instead of separately processing each one. If the exponent is divided in groups each *e* bits wide, the corresponding algorithm is characterized by the following features:

- the number of steps is $\lceil k/e \rceil$;
- at each step, the factor which multiplies the partial result is looked-up in a sub-table 2^e rows wide (chosen according to the step number) using the

Exponent bit	*k-1*	*k*-2	...	4	3	2	1	0
Factor	*B*^(2^*k-1*)	*B*^(2^*k-2*)	...	B^{16}	B^8	B^4	B^2	*B*

(a)

Exponent bit	*k-1*	...	...	(*s*+1)·*e*-1	...	*s*·*e*+1	*s*·*e*	...	2*e*-1	...	*e*+1	*e*	*e*-1	...	1	0
Step number	⌈*k*/*e*⌉-1			*s*				...	1				0			

Value of the bit group		step *s*		step 1	step 0
0 (unused row)		1		1	1
1		*B*^(2^*se*)		*B*^(2^*e*)	*B*
2	...	*B*^(2^*se*·2)	...	*B*^(2^*e*·2)	B^2
3		*B*^(2^*se*·3)		*B*^(2^*e*·3)	B^3
...					...
2^*e*-1		*B*^(2^*se*·(2^*e*−1))		*B*^(2^*e*·(2^*e*−1))	*B*^(2^*e*-1)

(b)

Fig. 2 - The precomputed base powers table construction
(a) simplest case, 1 bit per group
(b) general case, *e* bits per group

current group of exponent bits as the index; of course the factor corresponding to index 0 is always 1, thus it is not actually stored and the product is not needed;

- the table size is $\lceil k/e \rceil \cdot (2^e - 1)$ rows, because each group needs its own sub-table (Fig. 2b);
- if the exponent is random, the probability of a group of being zero is 2^{-e}, so the average number of modular products is $\lceil k/e \rceil \cdot (1 - 2^{-e})$.

The aforementioned straightforward solution can be obtained from this general definition by choosing e=1. In this case each sub-table degenerates to a single value, corresponding to an element of the k-entry table cited in the first definition.

Fig. 3 reports the storage allocation requirements of the enhanced algorithm and the computational load gain over the standard one, in function of e. Even using the smallest values of e there are evident computational benefits at almost no cost in terms of table size. For values of e between 6 and 8 a good tradeoff is achieved, as the computational load is reduced up to ten times while keeping the table size within reasonable values. For greater values of e, the huge growth of the table size is not repaid by the marginal gain in terms of computational load.

Eventually, the daily computational load deriving from the optimized exponentiations is function of the number k of bits in the exponent, and the bit group size e. It can be expressed in terms of modular products as follows:

$$L_{queries} = Q \cdot \lceil k/e \rceil \cdot (1 - 2^{-e})$$

3) To authenticate all the statements, the OWA value is updated, timestamped and signed. Due to the extremely low probability of receiving two or more status update notices without queries in between, this operation is usually performed after each status update. It is important to notice, though, that even an unchanged OWA could be timestamped and signed as often as desired, if keeping the timestamp fresh is considered more important than saving computations.

The digital signature operation is considered equivalent to a generic, non-optimized modular exponentiation, which, using the square-and-multiply algorithm as a reference, is equivalent to 1.5k modular products. The daily computational load, expressed in modular products, is then given by:

$$L_{authentication} = 2\frac{NP}{365} \cdot \left\lceil \lceil k/e \rceil \cdot (1 - 2^{-e}) + 1.5k \right\rceil$$

The sum L_{ORS} of the three contributions L_{update}, $L_{queries}$ and $L_{authentication}$ holds the overall daily computational load deriving from the application of the ORS scheme. Adopting a common metric, with all computational loads expressed in modular products involving 1024-bits numbers, L_{ORS} can be compared with the daily load deriving from the application of the OCSP scheme, which is easily expressed as:

$$L_{OCSP} = Q \cdot 1.5k \ .$$

A graphical representation of these quantities in function of the PKI population (Fig. 4) clearly shows the

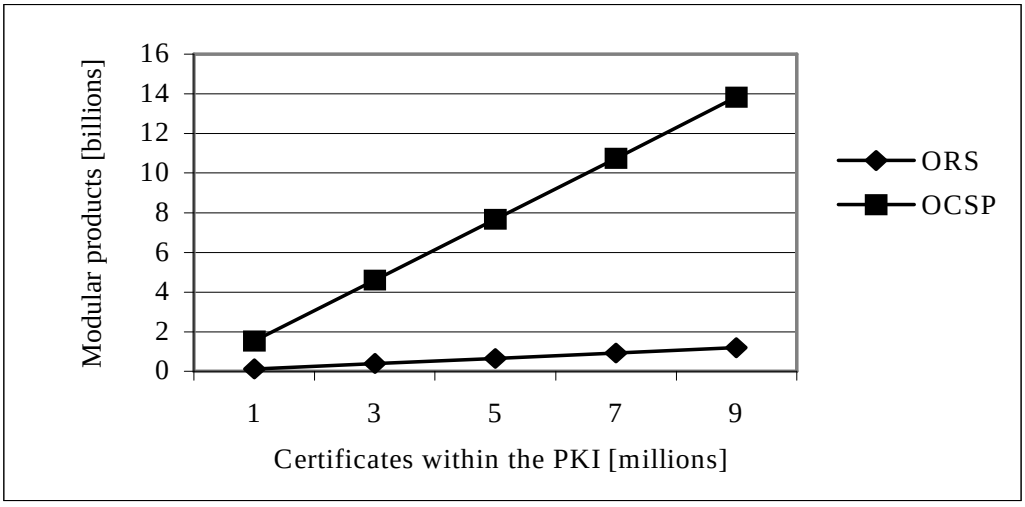

Fig. 3 - Performance of the enhanced modular exponentiation algorithm (k=1024)

load savings achieved by applying ORS. Yet more important, is the comparison between ORS and OCSP loads in function of the daily number of status queries (Fig. 5). As previously recalled, proper usage of a certificate should encompass real-time status verification. ORS, exhibiting a lower load dependency by Q, is able to sustain much more frequent status verifications.

5. Conclusions

In this paper a valuable solution to the problem of certificate status handling within PKIs has been presented. The devised scheme, aiming to maximixe the revocation timeliness, relies on a trusted directory. The reduced computational load with respect to similar on-line schemes, the directory outgoing traffic independence from the number of certificates handled within a PKI, and the concise and not forgeable proof of each single certificate validity are the main advantages of the proposed method.

Present activity is directed to further investigate some aspects related to the implementation of the devised scheme, with the aim of experimentally validate its performance. In particular, the effects of caching policies, of different statistical distributions of the various events, and of different population distribution among CAs are being examined.

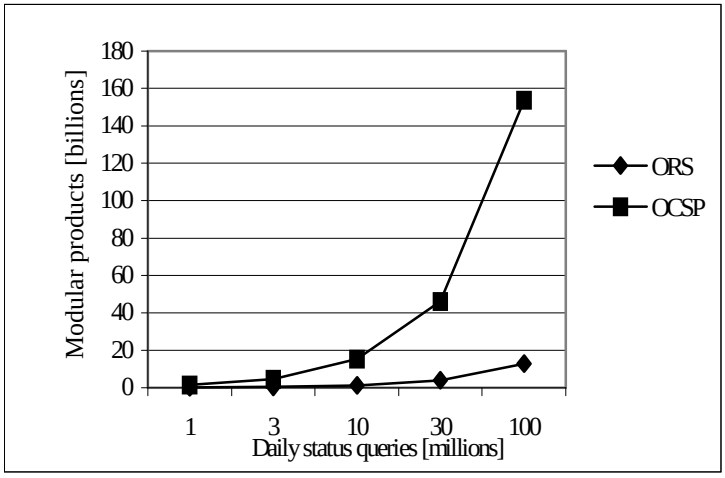

Fig. 4 - Directory computational load in function of N (k=1024, e=8, Q=N).

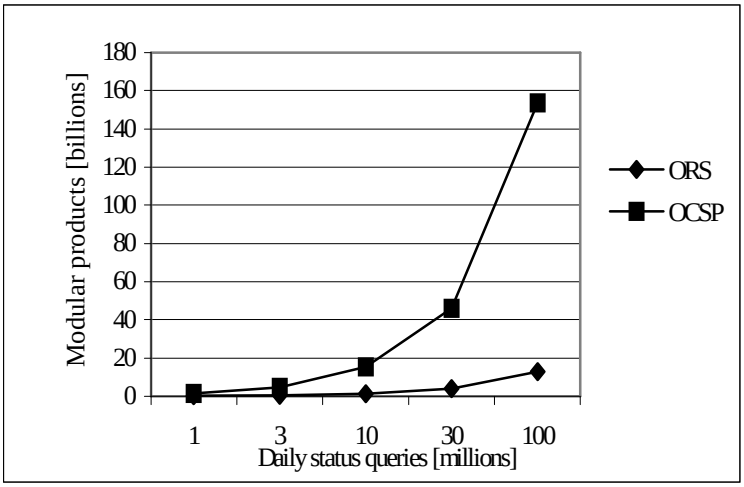

Fig. 5 - Directory computational load in function of Q (k=1024, e=8, N=3 millions).

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

[3] E. Faldella, M. Prandini: Efficient Handling of Certificates within Public-Key Infrastructures

a) Proc. 3rd IMACS/IEEE International Multiconference on Circuits, Systems, Communications and Computers (CSCC'99), Athens, Greece, July 4-8, 1999 (CD-ROM ISBN 960-8052-02-5).

b) "Computer and Computational Engineering in Control", ed. N. E. Mastorakis, World Scientific Engineering Society.

[4] M. Prandini, Efficient Certificate Status Handling within PKIs: an Application to Public Administration Services, Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99), Phoenix, Arizona, 1999 ("best student paper" award winner).

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

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

[7] M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, RFC 2560 – X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, The Internet Society, Jun. 1999, http://www.rfc-editor.org/rfc/rfc2560.txt.

[8] J. Benaloh, M. de Mare: One-Way Accumulators: A Decentralized Alternative to Digital Signatures (Extended Abstract). EUROCRYPT 1993 Proceedings, pp. 274-285.

[9] W. Diffie, M. E. Hellman: New directions in cryptography, IEEE Transactions on Information Theory, v. IT-22, n. 6, Nov 1976, pp. 644-654.

[10] W. Diffie, M. E. Hellman: Multiuser Cryptographic Techniques, Proceedings of the AFIPS National Computer Conference, 1976, pp. 109-112.

[11] R. Rivest, A. Shamir, L. Adleman: A method for Obtaining Digital Signatures and Public-key Cryptosystems. Communications of the ACM, v. 21, n. 2, Feb. 1978, pp. 120-126.

[12] B. Schneier: Applied Cryptography - Second Edition. John Wiley & Sons, 1996,pp. 95-96.

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

[14] NIST FIPS PUB 180-1, "Secure Hash Standard," National Institute of Standards and Technology, U.S. Department of Commerce, April 1995.