Using OCSP to Secure Certificate-Using Transactions in M-commerce

Jose L. Muñoz, Jordi Forné, Oscar Esparza, and Bernabe Miguel Soriano

Technical University of Catalonia (UPC)*
Telematics Engineering Department (ENTEL)
1-3 Jordi Girona, C3 08034 Barcelona (Spain)
Phone. +34934010804 Fax.+34934011058
{jose.munoz, jordi.forne, oscar.esparza, ibernabe, soriano}@entel.upc.es

Abstract. The possibility of making the Internet accessible via mobile telephones has generated an important opportunity for electronic commerce. Nevertheless, some deficiencies deter its mass acceptance in e-commerce applications. In order to speed up the information delivery, the use of brokerage systems constitutes an interesting solution. In this paper we review the problem of certificate validation in m-commerce transactions and we present an architecture where a broker is used as OCSP responder for the certificate validation. A modification over OCSP called $\mathcal{H}$-OCSP is also proposed as a way to reduce the computational load and the bandwidth requirements of OCSP which is specially desirable in the wireless environment. The ASN.1 add-on for $\mathcal{H}$-OCSP that makes it inter-operable with the standard OCSP is defined and the behaviour of $\mathcal{H}$-OCSP compared to standard OCSP is evaluated.

Keywords: broker, m-commerce, certification, certificate status checking, OCSP

1 Introduction

The access to the Internet by means of mobile devices potentially increases the number of users of e-commerce. One of the novelties of m-commerce is the possibility of attracting clients in the neighborhoods of commercial and/or service centers by providing them with appropriate information.

A way to ease and make more efficient the access to information from mobile terminals is to use a broker between the terminal and the wired network. Figure 1 shows a possible broker-based architecture. The broker re-uses the information and sends useful data in a predictive mode to wireless users, reducing data traffic in the wireless link.

* This work has been supported by the Spanish Research Council under the project DISQET (TIC2002-00818).

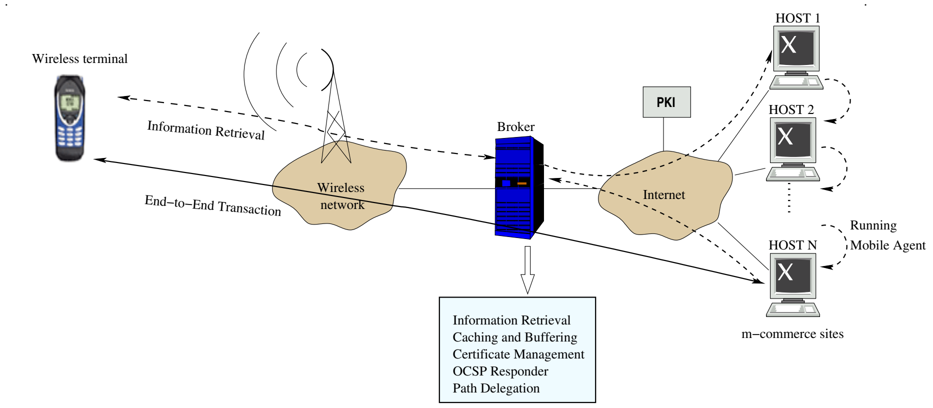

Fig. 1. Broker Architecture for a Wireless Scenario

The main functions of the broker are detailed below:

– *Intermediate storage*: the broker stores a copy of the information that it receives. This will allow its re-use by other users. Therefore, the information must be organized inside proxy and cache systems.
– *Customer Relationship Management infrastructure*: the knowledge of the user profile allows to manage the user's needs for information, and to deliver information in a predictive fashion and dynamic tracking of clusters of data. A user profile could be managed and updated dynamically, processing information from different sources. In wireless environment, CRM (Customer Relationship Management) can profit from the facilities of the push mechanism of mobile communication protocols.
– *Information search and retrieval*: tasks such as searching, advising, contacting, comparing, filtering and facilitating access to databases perfectly fit for mobile agent technology. In this sense, agents can perform some tasks on behalf of a user while the mobile device remains off-line, which is very desirable on a noisy weak link with unpredictable disconnection.
– *Certificate management*: a Public Key Infrastructure (PKI) is required to provide the m-commerce transactions with the security services such as integrity, privacy, non-repudiation, authentication and access control. The broker is appropriate for storing and managing digital certificates, and combined with OCSP (On line Certificate Status Protocol) [8] timely information about the status of certificates can be obtained. The rest of the paper deals with this function.

Digital certificates are usually used to identify parties involved in e-commerce and m-commerce transactions, as well as to provide end-to-end security in these transactions. A certificate is a digital document signed by a Trusted[1] Third

[1] Trust in a principal can be defined as a belief that the principal, when asked to perform an action, will act according to a pre-defined description. In particular, this

Party (TTP) called "issuer". Certificates are tamper-evident, in other words, they can be easily read but they cannot be modified without making the signature invalid. Moreover, they are unforgeable because only the issuer can produce the signature. There are several kinds of certificates but the most widely used are Identity Certificates (ICs), whose main function is to bind a public-key with an identity. An IC states an association between a name called a *Distinguished Name* (DN) and the user's public-key. Therefore, the authentication of the certificate relies on each user possessing a unique DN. DNs use the X.500 standard [11] and are intended to be unique across the Internet. On the other hand, the X.509 standard [5,12] defines what information can go into a certificate and its data format. All X.509 certificates have the following data, in addition to the signature:

– *Version.* This field identifies which version of the X.509 standard applies to this certificate, which affects what information can be specified in it. So far, three versions have been defined.
– *Serial Number.* The entity that created the certificate is responsible for assigning it a serial number to distinguish it from other certificates it issues.
– *Signature Algorithm Identifier.* Identifies the asymmetric algorithm used by the issuer to sign the certificate.
– *Issuer Name.* The DN of the issuer.
– *Validity Period.* Each certificate is valid only for a limited amount of time. It is not valid prior to the activation date and it is not valid beyond the expiration date.
– *Subject Name.* The DN of the entity whose public-key the certificate identifies.
– *Subject Public Key Information.* This is the public-key of the entity being named, together with an algorithm identifier which specifies which public-key crypto system this key belongs to and any associated key parameters.

It has been made also a profile for the use of ICs in the Internet by IETF in [4]. However, before performing a transaction it is crucial that the user makes sure that all certificates involved are valid, in opposite case, the transaction should be rejected since the securing mechanisms rely on the trust of the underlying certificates.

The rest of the paper is organized as follows: in Section 2 we introduce the problem of certificate validation in m-commerce transactions. In Section 3 we propose $\mathcal{H}$-OCSP as a way to reduce the computational load in the broker. In Section 4 we define the ASN.1 add-on for $\mathcal{H}$-OCSP that makes it inter-operable with the standard OCSP. In Section 5 we evaluate the behavior of $\mathcal{H}$-OCSP compared to standard OCSP. Finally, we conclude in Section 6.

belief implies the belief that the principal will not attempt to harm the requestor independently of the way it fulfills the request [10].

2 Securing Certificate-Using Transactions

Wireless users ask for information or services in the Internet and their requests arrive to the broker that will obtain this information on behalf of the user. After receiving the information, depending on the required service, the user may perform a transaction with a selected m-commerce site. When a user wishes to establish a transaction with a particular m-commerce site, an end-to-end authenticated and private channel is required in order to transfer sensitive data such as a credit card number. Technologies based on ICs, like HTTPS/TLS [1] are being widely used for establishing this kind of secure channels. However, prior to perform a transaction using a certain IC, the user must be sure that the certificate is valid. It must be stressed that the certificate management is a heavy process and that the clients in our environment are resource-limited. For this reason, clients delegate to the broker the processing of the certificates. Notice that the broker is a TTP and it is in general not resource-limited, therefore it is appropriate for storing and managing certificates.

The validation of a certificate comprises two mechanisms: certificate status checking and certification path validation.

The certificate status checking is needed because ICs have a bounded life-time and it may be necessary to revoke an IC before its life-time ends. An IC may be revoked, according to [3], because of:

- The loss or compromise of the associated private key.
- In response to a change in the owner's access rights.
- A change in the relationship with the issuer.
- As a precaution against cryptanalysis.

The two main standards for certificate status checking are Certificate Revocation Lists and Online Certificate Status Protocol.

The *Certificate Revocation List* (CRL) is the most mature approach, it is part of X.509 since its first version [12] and it has also been profiled for the Internet [4]. A CRL is a digitally signed list of revoked certificates, where for each entry in the list the following information is stored: the certificate serial number, the revocation reason and the revocation date. The CRL header includes information about the version, the CRL serial number, the issuer, the algorithm used to sign, the signature, the issuing date, the expiration date and some optional fields called extensions. The CRL is usually distributed to relying parties through non-trustworthy servers called "repositories".

The *Online Certificate Status Protocol* (OCSP) has been proposed by the PKIX workgroup of the IETF [8]. OCSP enables certificate-using applications to determine the revocation state of an identified certificate. The status of certificates is available online through trustworthy servers called "responders" that sign online each response they produce. An OCSP client issues a status request to an OCSP responder and suspends acceptance of the certificate in question until the responder provides a response. For e-commerce, online technologies are interesting not only because they can provide the status data in real-time but also because of billing issues (requests can be used as the basis for billing).

On the other hand, the Certification path validation is necessary when a relying party wants to validate a certificate of an end entity from a different issuer. A certification path is a chain of certificates where the issuer of the first certificate is a trustworthy entity (for example, the issuer of the relying party), and the subject of the last certificate is the end entity. Then, the user needs a mechanism to build and validate this chain of certificates. The problem of verifying a certification path remains an open topic (see "Simple Certificate Validation Protocol (SCVP)" [6] and "DPV and DPD over OCSP" [9]). This study is only focused on the certificate status checking mechanism.

Previous work by the authors have shown that CRLs are not a good choice for distribution of status data in resource-constrained handsets as it is the wireless link because of their high requirement in bandwidth and cache [7]. Therefore, the use of CRLs should be restricted to distribution of data among intermediate entities (e.g. distribution between the issuer CA and the broker). On the other hand, OCSP seems a good choice because a user can retrieve timely status data with a moderate resources usage. Moreover, if there is a need for transaction-specific authorization information, OCSP can provide it by means of its protocol extensions.

3 $\mathcal{H}$-OCSP: Secure and Efficient Status Checking

3.1 Preliminaries

As mentioned earlier, usually an OCSP responder signs online each response it produces. Responses can contain three times in them:

– `thisUpdate` is the time at which the status being indicated is known to be correct.
– `nextUpdate` is the time at or before which newer information will be available about the status of the certificate.
– `producedAt` is the time at which the OCSP responder signed this response.

The client may also send a nonce in its request. In this case, the responder has to include the nonce in the signature computation, and thereby the request and the response are cryptographically binded. However, a denial of service vulnerability is evident with respect to a flood of queries. The production of a signature significantly affects response generation cycle time, thereby exacerbating the situation. Unsigned error responses open up OCSP to another denial of service attack, where the attacker sends false error responses.

In order to alleviate these denial of service vulnerabilities, the OCSP responders may pre-produce signed responses specifying the status of certificates at a certain time [8]. The time at which the status was known to be correct shall be reflected in the `thisUpdate` field of the response. The time at or before which newer information will be available is reflected in the `nextUpdate` field, while the time at which the response was produced will appear in the `producedAt` field of the response. However, the use of precomputed responses allows replay attacks in which an old (good) response is replayed prior to its expiration date but after the certificate has been revoked. Deployments of OCSP should carefully evaluate the benefit of precomputed responses against the probability of replay attacks. In this sense, notice that it exists the following trade-off:

> *The online signature consumes much processing time. To reduce the possibility of falling into denial of service, the responder may pre-compute the responses and store them in a cache. But pre-produced responses are susceptible of generating reply attacks. To avoid the replay attacks, the responder needs to generate pre-produced responses within a short period of time which consumes many processing resources and this fact may lead the responder again to denial of service.*

3.2 $\mathcal{H}$-OCSP Basics

The result of the previous discussion is that the responder would benefit from a mechanism to pre-produce responses with low processing resources utilization. Below we outline a mechanism that reaches this target. Furthermore, under certain circumstances, the exposed mechanism has also many important benefits for the wireless clients that use OCSP.

The mechanism that we propose is called $\mathcal{H}$-OCSP and it exploits the fact that a OWHF (One Way Hash Function) is at least 10,000 times faster to compute than a digital signature. When an pre-produced response needs to be updated because its `nextUpdate` has become obsolete, a OWHF is performed to update this response instead of a new signature. Using an OWHF will permit the repository to update the responses more frequently without falling into denial of service.

$\mathcal{H}$-OCSP is based on the Even et al. algorithm [2] and it works as follows: when a response is going to be pre-produced, the responder adds a hash-chain to it. The hash chain permits the repository to update the pre-produced response in successive periods with a scarce resources utilization. The hash chain results from applying $d+1$ times a OWHF h over a secret nonce (1)

$$R \xrightarrow{h} R_d \xrightarrow{h} R_{d-1} \xrightarrow{h} \cdots \xrightarrow{h} R_i \xrightarrow{h} \cdots R_2 \xrightarrow{h} R_1 \xrightarrow{h} R_0 \tag{1}$$

Let us define the parameters involved in the process:

`primaryUpdateValue` (R) is the secret nonce. R is only known by the responder (broker) and it is generated for each new pre-produced response.

`maximumUpdateIndex` (d) is the maximum number of periods that a pre-produced response can be updated.

`baseUpdateValue` (R_0) is the last value of the hash chain and it is included in the signature computation of the pre-produced response. R_0 is computed by applying $(d+1)$ times h over R

$$R_0 = h^{d+1}(R) \tag{2}$$

`currentUpdateValue` (R_i) is computed by applying $(d+1-i)$ times h over R

$$R_i = h^{d+1-i}(R) \tag{3}$$

Where i is the number of periods "Δ" elapsed from the documented one (the documented validity period is the period included in the response). Δ is defined as

$$\Delta = \texttt{nextUpdate} - \texttt{thisUpdate} \tag{4}$$

A relying party can verify the validity of a pre-produced response that it is living beyond its documented life-time, say, at time t, where t is included within the period $[\texttt{nextUpdate} + (i-1)\Delta, \texttt{nextUpdate} + i\Delta]$, by checking the equality of equation (5)

$$R_0 = h^i(R_i) \quad with\ i \leq d \tag{5}$$

It must be stressed that to forge a `currentUpdateValue` with the information provided by a previous update value an attacker needs to find a pre-image of a OWHF which is by definition computationally infeasible.

It is obvious that if a pre-produced response becomes stale because the status of the certificate it refers changes, a new signature must be performed, in other words, $\mathcal{H}$-OCSP is only applicable to responses that need to update their documented life-time. However, notice that the majority of the response updates are due to the fact that the documented life-times become obsolete, hence the $\mathcal{H}$-OCSP will have indeed a great impact over the responder performance.

Notice also that $\mathcal{H}$-OCSP produces the desired behavior of pre-produced responses:

For one thing, the responder can use pre-produced responses with a small life-time which reduces the risk of replay attacks. For another, the repository can update its pre-produced responses at low cost which reduces the risk of denial of service.

Finally, it is worth mentioning that when the responder recovers from a crash or is restarted after being down for some reason, the most secure and easiest way to implement the startup environment is to erase all the pre-produced responses from cache. This is the way we perform the startup environment in our $\mathcal{H}$-OCSP responder. However, other implementations may keep the pre-produced responses but do so at their own peril.

Next, we show how to save bandwidth between the client and the responder using $\mathcal{H}$-OCSP. This feature is very interesting in the wireless environment where one of the biggest constrains is the scarce bandwidth that it is available.

Pre-produced responses can be also cached by clients. Let us assume that a previous $\mathcal{H}$-OCSP response for a certain certificate is stored in the client's cache. Then, if the client needs to check the status of the same certificate later, she can ask the responder for a `currentUpdateValue` instead of downloading a standard OCSP response which has a much bigger size. Moreover, if a client performs the majority of his requests for a small set of certificates while other certificates are more rarely requested, it becomes likely to have cached responses for these frequently asked certificates. In this case, the status checking can be performed only sending the `currentUpdateValue` and the best performance of $\mathcal{H}$-OCSP related to bandwidth utilization can be clearly appreciated (see Section 5).

On the other hand, a client may wish to keep an OCSP response as a consumer protection issue. If a client performs an important or a delicate transaction, it may later arise a conflict about the content of the exchanged messages. Usually, digital signatures performed during the transaction are used to solve these kind of non-repudiation trials, therefore it is necessary to have a proof of the status of the involved certificates in the precise moment that the transaction was performed. Notice that the hash chain permits to store old responses for a certain certificate with a reduced storage capacity. This storage can be performed by the broker or even by the client (for the certificates she considers important).

3.3 Security Discussion

Next, we present an informal discussion (rather than formal proofs or demonstrations) about the main security aspects of $\mathcal{H}$-OCSP.

Remember that OCSP can be used with:

– *Cryptographically binded requests and responses.* If the client wants a response cryptographically binded to her request, the repository must take into account the nonce that goes in the request when computing the signed response.
– *Pre-produced responses.* These responses are not bound to any particular request, so the information that a response contains can be re-used to respond as many requests as desired.

Cryptographically binded requests and responses are open to denial of service attacks because of the cost of computing the signatures. In this case the effectiveness of $\mathcal{H}$-OCSP avoiding this kind of denial of service attack fails for the following reasons:

– This mode of operation is very costly to the responder because it has to produce a different response per each client for the same information and store all these data in its cache.
– If the majority of the clients are honest and they collaborate with $\mathcal{H}$-OCSP by using their cache entries for previously asked data, the $\mathcal{H}$-OCSP responder will be more robust than the standard one. However, active adversaries can claim that they do not have previously asked for data (e.g. omitting their cache entries) and they can ask each time for a new response. In this case, the denial of service protection of $\mathcal{H}$-OCSP fails if attackers flood the responder with their requests ($\mathcal{H}$-OCSP will have the same asymptotic behaviour that OCSP).

As a result of the previous discussion, we discourage the use of $\mathcal{H}$-OCSP with cryptographically binded responses (or at least the administrator must be concerned about the risks of using this mode of operation).

On the other hand, when using pre-produced responses, the $\mathcal{H}$-OCSP responder is protected from denial of service attacks for data contained in its cache during (at the most):

$$\Delta_p = \Delta * d \tag{6}$$

We denote Δ_p as the "maximum protection period". Notice that any new request over data already contained in a cached response will not involve much processing capacity usage during the protected interval because the Even et al. algorithm can be executed almost in real time. Therefore the denial of service attack intended by the active adversary can be avoided.

Even though, the responder may have minor denial of service problems at start-up since at that time it must sign each produced response.

4 ASN.1 Add-on for $\mathcal{H}$-OCSP

In this section we address important implementation issues in order to make $\mathcal{H}$-OCSP inter-operable with the standard OCSP. In order to deploy $\mathcal{H}$-OCSP, we need to slightly modify the OCSP protocol. The additional parameters that we introduce are included in extensions. This provides compatibility because support for any specific extension is optional and unrecognized extensions are silently ignored by either the clients or the responders. The protocol is designed to permit inter-operability among standard OCSP clients and responders and $\mathcal{H}$-OCSP clients and responders with any combination among them. The ASN.1 add-on for $\mathcal{H}$-OCSP is presented in Figure 2.

An $\mathcal{H}$-OCSP client can include an extension in the `singleRequestExtensions` with OID `id-pkix-ocsp-base-update-value` to let the responder know that she understands $\mathcal{H}$-OCSP. If the $\mathcal{H}$-OCSP client has an $\mathcal{H}$-OCSP response for the target certificate in its cache, the extension includes the `baseUpdateValue`. Otherwise, the extension is filled with an array of 0 bytes. Upon receipt of a request, a responder determines if the message is well formed, if the repository is configured to provide the requested service and if the request contains the compulsory information.

If an $\mathcal{H}$-OCSP client requests a standard OCSP responder, the extension is silently ignored and the responder responds with a standard OCSP response. If an $\mathcal{H}$-OCSP responder receives a request, it looks for the correspondent extension. Depending on the request extension, the $\mathcal{H}$-OCSP responder will respond with different types of responses:

type-A is an $\mathcal{H}$-OCSP response that includes the `currentUpdateValue`. It is sent to the client if she understands $\mathcal{H}$-OCSP and if the `baseUpda- teValue` provided in the request extension matches the one currently stored by the responder.

type-B (basic) is a standard OCSP basic response. It is sent either if the client does not understand $\mathcal{H}$-OCSP or if it is the first request for a pre-produced response.

type-C contains a basic response plus a `currentUpdateValue`. The basic response includes also the `maximumUpdateIndex` parameter in one of the `singleExtensions` of the response. It is sent if the client understands $\mathcal{H}$-OCSP but it has not a previous $\mathcal{H}$-OCSP response in cache for the target certificate or if the cached response she has is obsolete (i.e. the `baseUpdateValue` does not match the one in the responder).

```
HOCSP DEFINITIONS EXPLICIT TAGS::=
BEGIN
IMPORTS

--PKIX Certificate Extensions
AlgorithmIdentifier, BasicOCSPResponse, id-pkix-ocsp FROM OCSP

AuthorityInfoAccessSyntax, GeneralName, CRLReason FROM PKIX1Implicit88
{iso(1) identified-organization(3) dod(6) internet(1) security(5)
mechanisms(5) pkix(7) id-mod(0) id-pkix1-implicit-88(2)}

Name, Extensions,Certificate, -- AlgorithmIdentifier, id-kp,
id-ad-ocsp FROM PKIX1Explicit88 {iso(1) identified-organization(3)
dod(6) internet(1) security(5) mechanisms(5) pkix(7) id-mod(0)
id-pkix1-explicit-88(1)};

--Type of H-OCSP responses
TypeAResponse ::= SEQUENCE { currentUpdateValue OCTET STRING }
TypeCResponse ::= SEQUENCE {
  basicResponse BasicOCSPResponse,
  typeAResponse TypeAResponse }
baseUpdateValue ::= OCTET STRING
maximumUpdateIndex ::= INTEGER

-- Object Identifiers (proposal)
id-pkix-hocsp-type-a OBJECT IDENTIFIER ::={ id-pkix-ocsp 8 }
id-pkix-hocsp-type-c OBJECT IDENTIFIER ::={ id-pkix-ocsp 9 }
id-pkix-hocsp-base-update-value OBJECT IDENTIFIER::={ id-pkix-ocsp 10 }
id-pkix-hocsp-maximum-update-index OBJECT IDENTIFIER ::=
{id-pkix-ocsp 11}
```

Fig. 2. ASN.1 add-on for $\mathcal{H}$-OCSP

5 Efficiency Comparison: OCSP vs. $\mathcal{H}$-OCSP

In this section, the authors compare the performance of standard OCSP versus $\mathcal{H}$-OCSP in terms of the down-link bandwidth consumption (responder-to-clients) and the processing capacity utilization in the responder.

The evaluation results have been obtained using our Java implementations of the standard OCSP and the $\mathcal{H}$-OCSP[2]. We have also checked inter-operability of our clients and responders with:

– Euro PKI (*http://ocsp.europki.org\:8026*).
– Open Validation (*http://ocsp.openvalidation.org\:80*).
– Alacris (*http://ocsptest.alacris.com\:3080/ocsp*).

[2] The software for the client and the server can be downloaded from `http://isg.upc.es/cervantes`

The experimental results have been obtained under the following conditions:

– Random revocations and random expirations are used[3].
– The test is configured with a database dynamism of one revocation and one expiration per hour.
– The clients generate 2 status checking requests per hour following an exponential probability density function.
– The target certificates are randomly chosen.
– There are 10,000 clients.
– Each client has a certificate.
– There is an average of 10% revocation.
– It is assumed that it has a group of 10 frequently asked certificates that take the 50% of the status checking requests.
– Pre-produced responses are used (non-cryptographically binded requests and responses).

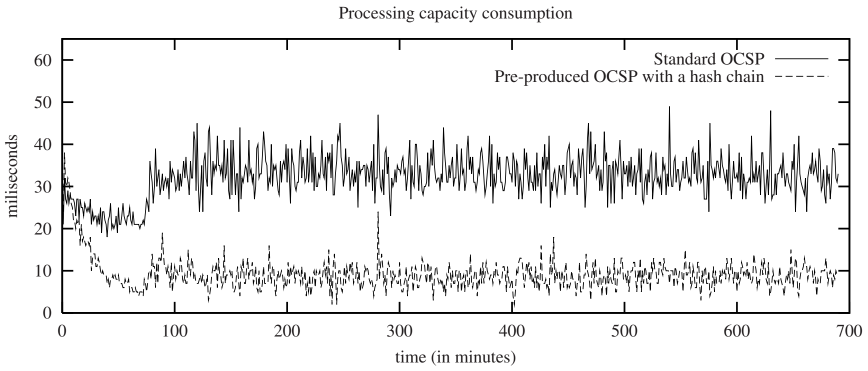

Fig. 3. Processing capacity consumption

Figure 3 shows the computational load of the responder. It can be observed that the computational load at the $\mathcal{H}$-OCSP responder decreases and becomes steady after a few minutes (approx. after 70 min, when the $\mathcal{H}$-OCSP responder starts to take advantage of the pre-computed responses).

In the steady state, the $\mathcal{H}$-OCSP responder consumes around five times less processing capacity than the standard OCSP responder. This is a substantial improvement taking into account that the CPU measurements include not only cryptographic operations but all the operations performed by the java virtual machine (notice that many of these operations such as the garbage collection are very time-consuming).

[3] A revocation implies adding a record to the database of the responder while an expiration implies removing a record from the database (it makes no sense to keep an expired certificate in the revoked certificates database of a responder). Notice

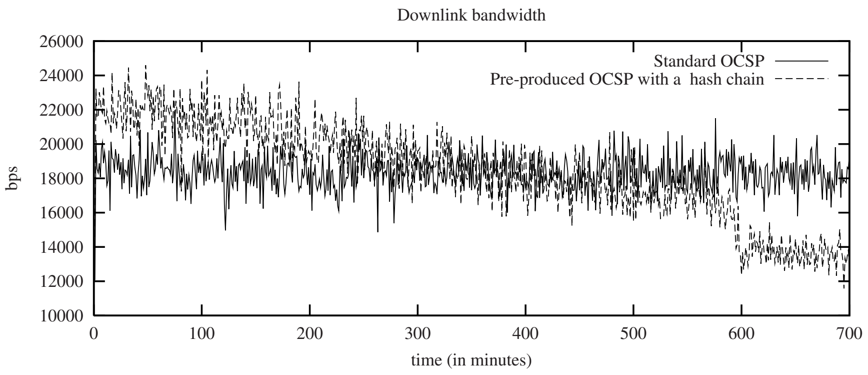

Fig. 4. Down-link bandwidth comparison

Figure 4 shows the measured down-link bandwidth utilization. In the steady state, when the client's cache is fully working, the better performance of $\mathcal{H}$-OCSP can be observed. With less percentage of requests for these frequently asked certificates the performance decreases but in the worse case is approximately as good as in standard OCSP.

6 Conclusions and Future Work

In this paper we introduce the problem of certificate validation in m-commerce transactions. We have shown that many of the validation functions can be delegated to a broker in order to alleviate the resources utilization in the client.

$\mathcal{H}$-OCSP has been proposed as a way to reduce the computational load of the responder. $\mathcal{H}$-OCSP is a hash chain-based mechanism to update pre-produced OCSP responses at a low cost. As a result, an $\mathcal{H}$-OCSP responder is better protected against denial of service attacks than a standard one. Wireless clients can also benefit from $\mathcal{H}$-OCSP by storing the responses of the most used certificates in their cache. However, there is a trade-off between the storing capacity of the wireless terminals and the benefits that $\mathcal{H}$-OCSP can achieve.

We are currently developing efficient cache updating policies for terminals with reduced storing capacity. On the other hand, we have defined the ASN.1 add-on for $\mathcal{H}$-OCSP that makes it inter-operable with the standard OCSP.

Finally, we have evaluated the behavior of $\mathcal{H}$-OCSP compared to standard OCSP and we have shown that $\mathcal{H}$-OCSP requires in general less resources. Although $\mathcal{H}$-OCSP has been designed with the wireless scenario in mind, some of its benefits also apply for the wired internet.

---

that each new record implies pre-producing a new signed response when a status request for this record arrives to the responder.

Acknowledgments. We would like to thank the Spanish Research Council for their support under the Grant TIC2002-00818. We also appreciate the valuable comments and contributions to this article made by the anonymous reviewers.

References

1. T. Dierks and C. Allen. The TLS protocol version 1.0, 1999. RFC 2246.
2. S. Even, O. Goldreich, and S. Micali. Online/offline signatures. *Journal of Criptology*, 9:35–67, 1996.
3. B. Fox and B. LaMacchia. Online Certificate Status Checking in Financial Transactions: The Case for Re-issuance. In *International Conference on Financial Cryptography (FC99)*, number 1648, pages 104–117, February 1999.
4. R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile, 1999. RFC 2459.
5. ITU/ISO Recommendation. X.509 Information Technology Open Systems Interconnection – The Directory: Autentication Frameworks, 2000. Technical Corrigendum.
6. A. Malpani, P. Hoffman, P. Housley, and T. Freeman. Simple Certification Validation Protocol (SCVP), December 2002. Internet Draft: draft-ietf-pkix-scvp-11.txt.
7. J.L. Muñoz and J. Forné. Evaluation of Certificate Revocation Policies: OCSP vs. Overissued CRL. In *DEXA Workshops 2002. Workshop on Trust and Privacy in Digital Business (TrustBus02)*, pages 511–515. IEEE Computer Society, September 2002.
8. M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP, 1999. RFC 2560.
9. Michael Myers. DPV and DPD over OCSP, January 2003. Internet Draft: draft-ietf-pkix-dpvdpd-00.txt.
10. P. Nikander. *An Architecture for Authorization and Delegation in Distributed Object-Oriented Agent Systems.* PhD thesis, Helsinki University of Technology.
11. CCITT Recommendation X.500. The directory overview of concepts, models and services, 1988.
12. ITU/ISO Recommendation X.509. Information technology Open Systems Interconnection – The Directory: Public Key and Attribute Certificate Frameworks, 1997.