ACVS: an Advanced Certificate Validation Service in Service-Oriented Architectures

Antonio Ruiz-Martínez¹, Daniel Sánchez-Martínez¹, C. Inmaculada Marin-López¹, Manuel Gil-Pérez¹, and Antonio F. Gómez-Skarmeta¹
¹Department of Information and Communications Engineering
University of Murcia, SPAIN
{arm, danielsm, inmaml, mgilperez, skarmeta}@um.es

Abstract

Proposals such as CAdES, XAdES and LTANS have made the need of remaining evidences over long periods of time clear. Between these evidences we have electronic signatures, certificates, requests and responses of validation mechanisms and so on. In order to gather these evidences, clients have to support several PKI-compliant protocols. Then, with the aim of simplifying this task XKMS appeared. However, XKMS only provides a simple validation mechanism that does not allow clients to specify some useful aspects needed for purposes of long term validation, such as trust anchors, information to store, responses to obtain, and so on. Therefore, in order to solve this problem, we present both an extension to XKMS and the different modules that are involved in this new specification.

1. Introduction

Electronic signature is an important aspect in electronic commerce since it provides some interesting features such as integrity, authentication and non-repudiation. For this reason, nowadays a great number of different platforms require the use of electronic signature and non-repudiation services. As a component of this kind of services, the certificate validation is one of the most important tasks to be performed. This task is responsible for checking the state of each certificate involved in the signature process until a trust anchor. This can penalize seriously the performance of some thin clients, like PDAs or mobile devices, which have limited resources.

Trying to shield applications from the complexity of the validation process, there is an emerging W3C recommendation called XML Key Management Specification (XKMS [1]), which specifies several protocols for registering, distributing and obtaining information about public keys. XKMS is based on Service-Oriented Architectures (SOA [2]) and defines a Validate service that is able to provide the validity status of a public key. However, the XKMS Validate service only returns a simple response without any additional information about the validation protocol or data used. This could be a disadvantage for applications that require the guarantee of an on-line validation to accept transactions, like electronic commerce services, and applications that need to recover and store this information in order to preserve a long-term validity of an electronic signature.

Recently, long-term validity of electronic signatures is a very important open issue [3,4,5]. There are several initiatives, like CMS Advanced Electronic Signatures (CAdES [6]) and XML Advanced Electronic Signatures (XAdES [7]), that work in the definition of structures to store electronic evidences records. The evidence records are usually attached to the electronic signed documents and are composed of some validation data and timestamps.

With the aim of offering advanced validation features that allow us to create these evidence records in a simple way, we propose the definition of an extension to XKMS Validate service. This proposal is called Advanced Certificate Validation Service (ACVS) and supports different kinds of clients, different validation mechanisms and several administrative domains. ACVS is able to produce the data structures that ensure the long-term validity of the electronic signatures to store them and to provide them later, if necessary, as electronic evidence records.

This paper is organized as follows. In section 2, we describe background information about current mechanisms for validating certificates. In section 3, the main goals and requirements of this proposal are specified. Section 4 introduces the system architecture and the main ACVS modules. In section 5, we explain in detail the main features of the validation service. Finally, section 6 summarizes the contributions of this paper and introduces some possible open issues.
2. Related Work

The process of validating a certificate consists of two steps. First, it takes place the construction of all candidate certification paths, starting with the certificate to be validated and ending with one issued by a trust anchor in which the relying party trusts. In a second step, it is carried out the validation of each certificate in each certification path discovered. This global process is referred to as certification path processing [8,9]. This is a complex process which could need the management of a large amount of information. With the aim of making it easier to the clients, several mechanisms have appeared to perform all or part of the validation process on behalf of them [3,10].

The first procedures and protocols used to perform validation processes were simple and incomplete. CRL [10] was a method totally offline and OCSP [10] performed such a process into a unique administrative domain. In the last few years, more sophisticated mechanisms have appeared to fix these limitations. Between them, we can find two standard protocols: Data Validation and Certification Server (DVCS [11]) and Server-Based Certificate Validation Protocol (SCVP [12]). Both of them allow a relying party to delegate certificate path construction and, optionally, certificate path validation to a trusted third-party, called Validation Service, according to a discovery and/or validation policy. These protocols make that every PKI can offer different mechanisms to perform certification path processing, thus making more complicated the work of the clients as they need to support them.

2.1. XKMS

XML Key Management Specification (XKMS [1]) is a W3C recommendation which is not tied to any specific protocol defined by the underlying PKIs, thus making the clients totally independent of the complexity that the knowledge of these protocols require. Clients can outsource the processing of key management to a dedicated server, by reducing substantially that complexity and avoiding the high processing and memory requirements that they need.

XKMS defines a Validate service, which may provide an assertion specifying the status of a given public key, depending on a single set of validation criteria. This service entails some disadvantages that should be taken into account [13]. On the one hand, it is not able to manage evidences for future non-repudiation checking operations that clients should need in advanced validation scenarios. On the other hand, a client is not able to specify which validation protocol should use the service to validate the certificates requested for him.

3. Main Goals for an Advanced Certificate Validation Service

The first goal that any advanced certificate validation service should fulfill is that clients should be as independent as possible from the underlying protocols used during the validation process. As second goal, the already existing validation mechanisms must be supported (OCSP, SCVP, etc.). Optionally, clients should be able to indicate which mechanisms the Validation service must use to make the certification path processing. The third goal is the support of different kinds of certificates belonging to several underlying infrastructures; at least, X.509-based and SPKI certificates. As fourth goal, it is necessary a repository where to store the validation evidences in for future non-repudiation checking operations (CRLs, OCSP, SCVP responses, etc.). As fifth goal, clients can specify the precise time in which the validation process must be carried out. Clients are also able to request if they want to obtain the evidences used during that validation process. The sixth goal is related to policy configuration. Clients must be able to select a specific policy to configure the behavior of the service. As a seventh goal, a control access configuration could be required to determine what entities can access the service and how they can do it.

Finally, the asynchronous procedures should be allowed since, sometimes, the responses cannot be delivered in that precise moment, and must be delayed until they are available.

4. Advanced Certificate Validation Architecture System

In this section, we describe the system architecture that we propose to cope with the extended validation problem and to satisfy the goals defined in the previous section. Next, we depict the elements and components that take part in the system architecture as well as the relationships between them. These components, shown in Figure 1, are: Public Key and Privilege Management Infrastructures (PKIs and PMIs), clients, XKMS server with the Advanced Certificate Validation Service (ACVS), and certificate validation protocols. ACVS, apart from checking the certificate status, provides some information that could be included in electronic signatures, as specified in CAdES and XAdES, which allows us to verify these signatures over long periods of time. Thereby, the service is responsible for building
the certificate chain (if it is required) and performing the validation process, returning, when necessary, the information used in that process. Additionally, ACVS could store some information used during the validation process.

Figure 1. Architecture system.

ACVS is able to use different protocols to build the certification path and verify the status of a certificate. These protocols are the ones provided by each PKI for verifying the certificates they issued.

The following sections present a detailed view of the main features provided by each component.

4.1. Public Key and Privilege Management Infrastructures

PKIs and PMIs are responsible for managing the complete lifecycle of identity and attribute certificates respectively. One of their most important functions is to provide information about the status of a certificate. Usually, attribute certificates are issued for short periods of time and, therefore, it is not necessary to check their validation status. However, these are sometimes issued for long periods of time, thus being required the provision of mechanisms to check them.

Although these infrastructures are mainly based on the use of X.509-based certificates, there exist other proposals of certificate format, such as SPKI certificates.

4.2. Clients

Clients request the status of certificates in a given moment in time to the certificate validation service. We find two kinds of clients: thin and thick clients. Thin clients, with limited resources, usually, only want to know whether a certificate is or was valid, and whether it satisfies a determined policy at a particular moment in time. In case these clients need additional information afterwards, they can request the server to store them. On the other hand, thick clients are not as limited as the previous ones and they can work, or they need to work, with additional validation information, but they are not interested in supporting all the different kinds of validation certification mechanisms. These clients could make use of this information, for example, to generate a signature according to the XAdES format [7].

Both types of client must be able to access the Advanced Certificate Validation Service by means of a XKMS Validate request. Additionally, they should support the extension that we have defined in order to get further information about the validation.

In this way, we maintain backward compatibility to current XKMS systems as well as we facilitate the migration to the proposed service.

The basic element in a validation request is the certificate. Optionally, the client can specify a time to validate the certificate in and a validation policy. If the time is not specified, the server validates it at the reception moment. If a validation policy is not specified, the default policy is then applied. Moreover, clients can specify how the server will behave in the validation process. Thus, clients are able to select the validation mechanisms, or request additional validation information related to the validation process, such as requests and responses of the validation mechanisms used, the path of the certificate up to a trusted point, a timestamp, and so on.

4.3. Advanced Certificate Validation Service Modules

The Advanced Certificate Validation Service (ACVS) is responsible for building certificate chains of and validating the certificates requested by clients. In order to perform these tasks, ACVS needs several modules. These modules are mainly based on SAVaCert [14], although we have considered some interesting ideas from ECPV [15], such as the harvester and scheduler modules. Then, in order to satisfy the different goals that this service should reach, we need to include new additional components. Figure 2 depicts all the components included in this service.

The authorization module is responsible for allowing the user to gain access to this service. This access could be made without any restriction, so this module would not be necessary. However, this service could be offered by a PKI or an enterprise that only wants to authorize its users to access it. This module is also useful in those situations where the access to the
service is based on a commercial agreement between a client and the entity that provides the service. In this last case, we would need an additional module responsible for the accounting. This accounting module is in charge of measuring the number of resources that a user or entity consumes during the access. This could include the number of validations made or the amount of information stored.

The asynchronous request module is responsible for managing those requests that cannot be answered immediately, such as when these require the building and validation of complex certification paths. In these cases, the server stores the request to be processed later on, and sends a response to the client to inform him that he could check later if the response is available. When the user requests information about the status of the operation, he could get two possible answers: a response indicating that he should check the result later or an answer with the response to his original request.

The user’s profile module allows users to choose the profile they want to be applied to their requests. A profile indicates which information should be included in requests and responses.

The service policy management protocol manages the different policies that could be applied during the execution of the protocol. A policy specifies how the service has to work regarding to a specific profile.

The harvester and scheduler modules are, according to the ECPV architecture, in charge of storing information related to the different certificates, with the aim of accelerating future requests.

The evidences store module is used to store the different electronic evidences related to a specific validation request, in case the requester needs them later.

Finally, a log module has been included to registry the different events that occurs in the architecture so as to detect possible attacks, problems related to availability, implementation errors and so on.

4.4. Certificate Validation Protocols

In order to gain access to the ACVS, clients use the SOAP protocol. The requests and responses to the service are built according to the XKMS specification, as well as the extensions to this specification that we have defined. Then, the service uses some mechanisms and protocols to build the validation chain and make the subsequent validation. The protocols used by the service depend on the ones offered by every PKI, which could vary from on-line validation mechanisms (OCSP, SCVP, etc.) to off-line ones (CRLs). Therefore, the users only have to support the access to the service we propose while this service is responsible for supporting the different underlying protocols.

Additionally, our proposal allows the user to specify the protocols that the service should use to accomplish the validation process. This functionality is essential in some scenarios, like electronic commerce, where specific validation protocols are required (in some e-commerce scenarios, it is required the use of on-line mechanisms instead of off-line mechanisms).

Moreover, this new service could also access to other PKIs that implement a XKMS server, since the number of PKIs offering this service is increasing. Even more, this new service could invoke another service that provides the advance validation service that we propose.

5. Advanced Certificate Validation Service Specification

This section presents the service proposed and how clients can make request to it. According to Figure 1, this service is accessed by both thin and thick clients. Furthermore, a system architecture that supports this service (XKMS server with ACVS) could use this protocol to get information from other systems or PKI infrastructures that support it (see PKI XKMS+ACVS in Figure 1). The service is based on the extension of XKMS Validate service. We have extended Validate service because it does not allow achieving the goals defined in section 3.

5.1. XKMS Validate Service Extension

In the XKMS Validate service, two kinds of messages are used: ValidateRequest and ValidateResult. These messages extend the MessageAbstractType which contains the element MessageExtension that was conceived to extend XKMS messages.

With the aim of satisfying the goals in section 3, for each message we have defined two new elements that
could be included in the MessageExtension field: ExtendedValidationRequest for the ValidateRequest message and ExtendedValidationResult for the ValidateResult message.

With this extension, if a client wants that the service validates a certificate, he has to send it a ValidateRequest message. The information about the certificate to validate is provided by means of the QueryKeyBinding element of ValidateRequest. The time the certificate is validated in is, optionally, indicated in the TimeInstant element of the QueryKeyBinding. Additionally, the client can fill in the MessageExtension field within the ExtendedValidationRequest element that we have defined. This element is composed of some components that allow the inclusion of parameters related to the validation process and the information to be retrieved and archived, such as the specific protocols to make the validation with, the information to indicate that the service should make the complete path validation, if the certification path should be returned in the response and the information to store so as to use it in a subsequent moment with the aim of proving the existence of such evidences. These evidences could be stored in a repository within the service, or it could use a specific server according to LTANS architecture [4,16]. It is important to point out that this element has been defined to be compatible with LTANS information in order to have both possibilities. Furthermore, in this structure we have defined an element, called OptionalInputs, for extensibility purposes. The access to the service could be authenticated by means of HTTPS or credentials, such as SAML Assertions or Archifacts.

As a response to a request, there are two kinds of responses. In the first one, the service could only send a simple response indicating if the certificate is valid or not at the moment specified (when the ExtendendValidationRequest is not included in the request). In the second one, the service could send a more elaborated one with the complete path certificate chain, the answers returned by the mechanisms used and so on (when the extension field is included). The service could also store the validation information that the client requested. This is useful when the resources of clients are limited or when we want a trusted third-party which offers an archiving service to store the information used during the validation process. The parameters indicated in the request could be simplified by means of the specification of a policy and a profile. The response is sent in a ValidateResult according to the XKMS Validate service. If the request contained the extension, then the response must contain the element called ExtensionValidationResult within the MessageExtension field.

The ExtendedValidationResult element is used to include all the additional information about validation asked in the request message with the ExtendendValidationRequest. Here, we have defined three elements. The first one indicates the validity of the certificate and the information related to its validation (CertificateValidity). The second one includes the information about the path of each certificate (CertificationPath). And the third one is used for extensibility purposes (OptionalOutputs). Finally, the response message could be authenticated by means of a signature.

Figure 3. Messages in the asynchronous mode.

The aforementioned process is called the synchronous mode because the response can be provided (almost) immediately. However, sometimes the response cannot be provided immediately. In this case, the asynchronous mode is used. In this mode, see Figure 3, the request is the same as in the synchronous mode. However, the server answers with a ValidateResult, with the value PendingRequest as result of the validation. This indicates to the client that he has to wait for some time to get it. In this situation there are two possible alternatives. First, the client can subsequently use a StatusRequest message to check whether the response is available. This step should be executed several times until the response is available. The other possibility is that the server informs the client that the response is available. Once the client is aware of the response is ready, he sends a PendingRequest message so as to get it. The response is received in the ValidateResult message which contains an ExtendedValidationResult.

6. Conclusions and Future Work

The certificate validation process is an important task in electronic processes where electronic signature and non-repudiation services are involved. In order to simplify the certificate validation process to clients,
XKMS specification appeared. However, in some scenarios where it is required that the electronic signature can be verified over long periods of time, it is needed that the validation process returns some evidences to clients. This capability is not supported by XKMS specification. With the aim of solving this problem, we have extended XKMS specification in order to provide an advanced validation service and the different modules that are needed to support that functionality.

The Advanced Certificate Validation Service proposed supports different kinds of certificate status checking protocols, the possibility of checking certificates from multiple PKIs, the configuration based on policies and profiles, control access based on the client’s credentials, the retrieving and storing of additional information about a certificate validation process. It is also adaptable to different kinds of clients, such as thin clients, e-commerce clients, e-government clients, PKI clients and so on.

As regards future work, we are considering the designing of a whole long-term signature archiving architecture that includes this proposal as an important component. Another aspect to be taken into account is the trusted relationships that a validation service could have with other validation services.

7. Acknowledgements

This work has been funded by ECOSPACE “Integrated Project on e-Professional Collaboration Space” (FP6 IST-035208) and Programa de Ayuda a los Grupos de Excelencia de la Fundación Séneca-Agencia de Ciencia y Tecnología de la Región de Murcia.

8. References

[16] IETF. Long-Term Archive and Notary Services (LTANS), IETF Working Group.