J. Lopez, A. Mana, J. A. Montenegro, and J. J. Ortega, "PKI Design Based on the Use of On-line Certification Authorities", *International Journal of Information Security (IJIS)*, vol. 2, pp. 91-102, 2004.
NICS Lab. Publications: https://www.nics.uma.es/publications

International Journal of Information Security manuscript No.
(will be inserted by the editor)

PKI Design Based on the Use of On-Line Certification Authorities

Javier Lopez, Antonio Maña, Jose A. Montenegro, Juan J. Ortega

Computer Science Department, University of Malaga
Campus de Teatinos, 29071 - Malaga, Spain
e-mail:{jlm, amg, monte, juanjose@lcc.uma.es}

Received: date / Revised version: date

Abstract Public-Key Infrastructures (PKIs) are considered the basis of the protocols and tools needed to guarantee the security demanded for new Internet applications like electronic commerce, government-citizen relationships and digital distribution. This paper introduces a new infrastructure design, Cert'eM, a key management and certification system that is based on the structure of the electronic mail service and on the principle of near-certification. Cert'eM provides secure means to identify users and distribute their public-key certificates, enhances the efficiency of revocation procedures, and avoids scalability and synchronization problems. Because we have considered the revocation problem as priority in the design process, and with a big influence in the rest of the PKI components, we have developed an alternative solution to the use of Certificate Revocation Lists (CRLs), which has become one of the strongest points in this new scheme.

1 Introduction

Internet technology is growing at explosive rates, but it is still constrained by security issues. Public Administrations and commercial companies have adopted this technology in limited aspects; essentially, as an informational vehicle. Further progress in those important areas has been limited by the open design of the network itself [1], among other reasons. At the same time, actual technology provides means for the interception, monitoring and forging of messages, and even impersonation. Consequently, users are still quite reluctant to use the network as the channel to send/receive sensitive information.

The need for user authentication is clearly essential in many Internet applications. However, although different solutions have been proposed, the problem still needs to be solved in an optimal way. Several systems, such as Kerberos were proposed in the past to protect communications over public networks using symmetric-key cryptography [2,3]. Those systems have shown scalability difficulties for the case of users belonging to a large number of different organizations. Even later efforts intended to solve that problem have not been successful [4–6].

Solutions based on public-key cryptography [7] have been shown to be well suited to satisfy the requirements of the Internet, becoming the foundation for those applications that require security and authentication in open networks. Certainly, the widespread use of a public-key cryptosystem requires a *Public-Key Infrastructure* (PKI), an efficient and trustworthy mean to manage a large number of users' public-key values. A PKI is a vital element because it provides confidentiality, integrity, authentication and non-repudiation services. Thus, it ensures the security of electronic transactions, and the exchange of sensitive information between parties that do not have a face-to-face interaction. The reason is that it is impractical and unrealistic to expect that each user in a large-scale network has a previously established relationship with all other users. Without such a functioning infrastructure, public-key cryptography would be only marginally more useful than techniques based on secret keys.

Certification of public keys is the first basic function of a PKI. PKI users trust the *Certification Authorities* (CAs). These are trusted entities that create and assign *public-key certificates*, computer-based records that attest the connection of public keys to identified subscribers. CAs attach their own digital signatures to provide assurance to the authenticity and integrity of the certificates.

The second basic PKI function is certificate *validation*. The information signed by the issuer CA can change over time, but every user needs to be sure that all data in the certificate are trustworthy and up to date. A process related to validation is certificate *revocation*. Security of private keys is a major problem in public-key cryptography, but it is expected that users' keys may be lost or compromised. Additionally, when a company goes

out of business, or an employee quits, is fired or transferred to a new position, a key may no longer be needed or used. Thus, there are many reasons why a key may need to be revoked before it expires.

Revocation has been a major problem in most of PKI designs. There is no doubt that when this problem is not properly addressed the PKI solution becomes very inefficient. In this paper we present *Cert'eM* (Certification based on e-Mail address structure), a PKI scheme that has been designed to solve many of the problems associated with revocation procedures. Also, and as we will show in this work, the scheme provides other advantages: it is adapted to real application scenarios, and because it uses the multi-hierarchical Internet structure, the operation of the CA architecture satisfies the needs of nearby-certification and, it avoids scalability problems.

The remainder of the paper has the following structure. Section 2 presents the main considerations for the design of our scheme. To be more precise, we have elaborated on the problem of user identification and, as stated, on the problem of revocation procedures. Section 3 explains the general goals of Cert'eM, and shows how its structure and internal operation helps to achieve such goals. Section 4 describes the algorithm that we have developed for the determination of the physical location of KSUs, which are the main elements of the architecture. Section 5 describes in detail the protocol for accessing the Certificate Servers, which are the elements included in KSUs that receive certificate requests and deliver the certificates to those requests. Finally section 6 ends up with conclusions.

2 Main Considerations for the Design of the New Scheme

2.1 Authentication and identity of users

In the real world authentication is achieved in different ways, and identities are not intrinsically restricted to one per physical person [8]. The absolute authentication of the real-world person using a particular computer is not necessary for many computer systems. Instead, relative authentication is enough; i.e. the computer does not need to know the legal name of the person, but simply needs to verify that, for instance, that person is authorized to access the system. In this sense, authentication systems can be classified as follows [9]:

- *Password - based systems.* In these systems the user knows some information that no one else knows. When the user is requested, he/she provides all or part of the information to the verifier. The classical username - password schemes are the most widely used authentication systems because of their simplicity and ease of implementation. Their important properties are transitivity and not exclusiveness. Once a user discloses the secret information to another user, the latter cannot be distinguished from the first one and acquires the same capabilities; thus, the system will authenticate both users as the same one. The main problems associated with this type of systems are: (a) The password can be intercepted in the way to the destination computer; the attacker that learns the password can impersonate the owner; (b) passwords are frequently forgotten; so it is necessary to have a mechanism to establish a new password without knowledge of the previous one, and that introduces a serious security breach; (c) users normally select passwords easy to guess, and sometime share them.
- *Physical tokens.* A token is a physical object the user has and that, somehow, proves his identity. The most typical examples are access cards, which grant access to restricted areas, and smart cards. Physical tokens are transitive but also exclusive; when a user gives the token to another user this one gains full capabilities of use, and simultaneous authentication of both of them is not possible. The problems these systems present are: (a) The token does not really prove user's identity and, thus, anyone who gets possession of the token is positively identified; (b) if the token is lost or damaged the system will not be able to identify the user; (c) most tokens can be easily copied or forged.
- *Biometrics.* These systems use data extracted from some biologic characteristics of the user (iris image, fingerprints, voice, handwriting, etc.) using a device that is not likely to produce the same value for different persons. Oppositely to the previous types this scheme is not transitive and, therefore, exclusive. Biometrics can be a reliable way to establish user identity for applications like physical access control, but are not well suited for Internet user authentication. Main problems associated with these systems are: (a) Biometric profile of a user has to be stored in the computer before the user can be authenticated; (b) devices used to extract the biometric characteristics of the user are expensive; (c) devices are also vulnerable and special protection is required to avoid sabotage or fraud.
- *Location.* Several companies commercialise authentication systems based on the *Global Positioning System* (GPS). These systems have the same drawbacks as biometrics; they are expensive and exposed to threats related to the base equipment. There are devices (designed for military use to defeat missile navigation systems) that can forge satellite GPS signals.

All of the former authentication systems are well suited for small communities of users and, hence, we could call them 'private' authentication systems. However, the Internet is formed by a very large community of users. In order to guarantee secure user authentication for applications in a worldwide environment, and over

open networks, previous authentication systems are not applicable.

In the real world we have ways to establish the identity of a person (passports, national identity cards, driver licenses, etc). It is clear that digital certificates are the most similar mechanisms actually available to mimic the properties of these types of paper-based documents.

Recalling from previous definitions, if we want to securely identify Internet users we must find "something that makes users easy to recognize" (or "something that makes users different in some way"), and that is usable in the digital world. Our system, Cert'eM, uses the combination of the e-mail address and the cryptographic key of the user to make him/her different. This combination fulfils most of the criteria needed for human authentication.

During the last few years the research community has learnt about PKI schemes where issuing of identity certificates has been based on a contact, through Internet, between the user and the certifier, not conforming to legal requirements [10]. This is clearly unsatisfactory because the verifier of a certificate will usually require some proof of the link between the identity of the user in the real world and his/her name in the Internet world. Therefore, for certificates issued in such way, trust is misled from the start.

The design of Cert'eM guarantees that a CA only certifies the keys of closely related users. A formal identity verification procedure is established (in accordance with the applicable law) in order to give the certification process a legal meaning [11]. Therefore, when the CA signs, say, *Alice's* key, it guarantees that *Alice's* identity has been conveniently verified. Consequently, a link is established between the identity documents (valid in the real world), a distinguished name in the Internet world (the electronic mail address) and a cryptographic key.

There are two common criticisms regarding the use of e-mail addresses as distinguished names. Firstly, it is claimed that the relationship between a person in the real world and an e-mail address is not one-to-one because a user can own several e-mail accounts and different aliases. Besides, there are certain e-mail addresses that do not represent a single user but a group of them [12]). Secondly, it is also claimed that, in some cases, the alias file can be modified without administrator or root permissions.

Cert'eM is not exposed to any of these disadvantages because it makes no distinction between an e-mail account and an alias. Let us review the possible cases in detail. Regarding the first criticism:

- *Single User Alias*: Let's suppose that *Alice* wants to create an alias (e.g., *al@x.y.z*) for her initial e-mail address, *alice@x.y.z*. This alias will be linked to her name in the real world and to a cryptographic key. The key can be the same one that she is already using linked to her initial e-mail account or, alternatively, she can choose a new one. There is no difference to the system. Therefore, the relation <*registered address, real world user*> is one-to-one.
- *Group Alias*: If a group of users want to register an alias for their addresses, they have to follow the registration procedure and, naturally, the responsibility of sharing the private key relies on them. Hence, although the relation <*registered address, real world user*> is still multiple, there is a finite well-defined group of users related to that address.

Regarding the second criticism, and although a malicious user could change (under some circumstances), the address that is represented by an alias, no problem occurs unless the registered user of the alias passes his/her private key to the malicious one. It is clear that a user will not be able to certify and insert the new key in the certificate repository.

2.2 Certificate Revocation

Most of PKI models use a *Certificate Revocation List* (CRL) as the mechanism to implement certificate revocation. This method is defined by Recommendation X.509 [13], which is the recognized standard for public key certificate formats. In this Recommendation a CRL is defined as a time-stamped list, identifying revoked certificates, which is signed by a CA and made freely available. Each revoked certificate is identified inside a CRL by the certificate serial number.

The *pull method* for CRL distribution is the most common method to check the lists of revoked certificates. The CRL is not automatically distributed; on the contrary, users access the list that is periodically published by the CA. A major drawback of this method is that *time granularity* of revocation is limited to the CRL issue period. As figure 1 shows, significant time intervals can take place: (i) since a user decides to revoke the certificate until the CA is informed, (ii) from that moment until the new CRL is issued and, (iii) from here until the CRL modification is available to final users. During those periods, the risk of integrity in the system grows exponentially, and certainly, it is not clear who holds responsibility during each of them.

Another important limitation is the size the CRL can reach. If the repository becomes very large, performance problems are produced in terms of both communication and processing overheads. There are two elements where simultaneous growth produces difficulties in CRL management. The first one is the revocation rate, clearly dependent on the size of users population, and quite unpredictable. The second one is the certificates validity period, because once the CA adds a certificate into the CRL, it is not removed until the expiration date.

The first element - number of users - cannot be easily controlled. However, the second one - validity period - seems so because it depends on the certification policy

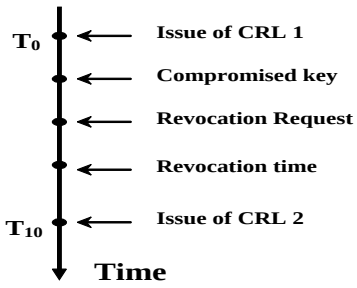

Fig. 1 Timeline of a certificate revocation process

established inside the PKI. Lowering the certificate validity period in the certificate creation process, the CRL becomes smaller (very small validity periods could even eliminate the need to issue CRLs). But, in this case, what is positive for CRL size is negative for the general performance of the system. The reason is that the use of small validity periods implies that certificates must be renewed more frequently. Therefore, this becomes a costly and ineffective solution and can be used only in some specific cases.

As modification of the validity period turns problematic, we need to analyse again the first element. Because the number of users cannot be decreased, the solution is to distribute them; that is, to partition the revocation repository into *CRL distribution points*, with each point containing a disjoint group of revoked certificates. In version 3 of X.509 [14] the concept of distribution point is extended from previous versions, allowing those points to be established depending not only on the subject type (final user or CA), but also on the reason of revocation. Furthermore, certification policy allows each list to be issued at different times. The X.509 v.3 also introduces the concept of *incremental CRL*, or *Delta-CRL*. According to this concept, a CA does not need to issue new periodic versions of a CRL; it just needs to issue the modifications (Delta) from the last version.

Our consideration is that all possibilities concerning the pull method represent a collection of difficult solutions. This method is far from the optimal solution that most of real PKI applications demand.

There is a less used method for CRL distribution, the *push method*. In this method, the CA sends the revocation lists to the users periodically, and does not introduce it into a repository as in the pull method. Such broadcast is accomplished via protected communication means, such as secure e-mail or a protected transaction protocol. The major advantage of this approach is that important revocations can be distributed very quickly, considering the time granularity problem inherent to the periodic revocation list approach. However, there are two potential problems with the push method. Firstly, the requirement for a protected distribution method that guarantees that CRLs reach the intended destinations. This represents an overload for the system. Secondly, the massive amount of traffic generated in order to notify all revocations.

Therefore, neither pull nor push methods seem to be optimal choices to solve generation, management and distribution of CRLs inside a PKI. In fact, if a certificate is not included in a CRL, then what the user knows is that the certificate was not revoked when the CRL was issued, but in most cases this is not enough. The reason is that CRL-based systems provide negative proofs (proofs of the negative validity) but they give no evidence of the positive validity.

We consider that the concept of CRL itself represents a drawback, and that the best solution is that one in which the knowledge of a revoked certificate is immediately available to users without a lack of performance in the system [15]. This idea has been followed in the design of our system, Cert'eM, where certificate revocation problem has been faced with special detail.

Usually, a positive proof of the validity of the certificate or, even better, a proof of the status of the certificate, is needed. In Cert'eM we use this type of proof, and call it *validity statement* (VS). For uses other than historical (i.e., knowing that the certificate was once issued), the requirement of previous possession of the certificate does not represent any advantage to obtain confidence in the information it includes. On the contrary, it introduces serious obstacles to the certificate management and use procedures. We believe that an *online certificate server* can solve the certificate validity problem more efficiently than CRL-based systems. After all, the CRL retrieval requires an online request.

Moreover, distributing the contents of the database of certificates among a series of servers according to some established distribution criteria is an even more efficient approach. In fact, a good distribution scheme should fulfil the following properties: (a) An algorithm exists that applied to the known data (key) of the certificate (not the complete certificate), unambiguously identifies the server that contains it; and (b) the scheme distributes the certificates in a balanced manner (i.e. the number of certificates stored in each server must be proportional to the capacity of the server). As we will see in next section, Cert'eM fulfils these characteristics.

3 Description of the System

3.1 Design Principles

The following list shows the fundamental design principles of Cert'eM. These goals have been established in order to keep a balance among flexibility, ease of use, efficiency and, of course, security:

- Adaptation to real application scenarios, that is, to multi-hierarchical Internet structure;

- Distinction between the real world (where we deal with concepts like "person","company","computer", etc.) and the Internet world (where we deal with concepts like "keys", "certificates", "names", etc.), in such a way that the trust used in the real world can be abstracted and computerized by using certificates;
- Provision of a secure mean to identify users and to distribute their public keys;
- Operation of a CAs architecture that satisfies the needs of near-certification so that trust can be based on whatever criteria is used in real life;
- Elimination of problems associated with the revocation procedures, and simplification of certificates validation;
- Avoidance of architectures that yield scalability problems; and
- Avoidance of the synchronization problems associated with schemes that keep multiple copies of the keys and certificates.

3.2 System Structure

The structure of our system is especially suitable for hierarchical organizations, which is the case of most private companies and Public Administrations. The main element in the hierarchy is the *Keys Service Unit* (KSU), which integrates both key certification and certificate management functions. Cert'eM uses a scheme with several KSUs operating over disjoint groups of users as shown in figure 2. The KSU hierarchy defined by Cert'eM is parallel to the hierarchy of Internet domains, or to be more precise, KSUs are associated with the corresponding Internet e-mail offices. Cross-certification is done at top-level nodes among different hierarchies.

As shown in figure 3, every KSU has three main components. We describe them in the following paragraphs.

- *Certification Authority.* The role of the CA is not only to issue users' certificates in its domain. At the same time, and as we will show later, it is involved in user registration procedure; that is, Cert'eM makes no use of *Registration Authorities*, and all related functions.
  A special user, that we denote as *CA@domain*, represents the CA of every domain (i.e., *CA@lcc.uma.es*). The CA itself is composed by two elements, the *Certification Kernel* (CK) and the *Certificate Server* (CS).
  The CK runs some of the tasks related to certification management in the CA, as key-pair generation (or the strength of those keys if generated by the user), as well as the creation and storage of digital certificates. Additionally, it runs tasks that are independent of certificate management, like those tasks related to the compilation of statistics regarding users' certificates requests.

The CS basically has two functions. The first of the functions is to receive certificates requests demanded by users of the local domain and to search for external ones (following a procedure that we will show later). Obviously, the second of the functions is to accept remote requests regarding local users' certificates demanded by other KSUs in the hierarchy, and to provide the service (delivery of certificates) to those requests. The service is supported by a number of processes, created in real time, that operate concurrently in order to achieve high efficiency.

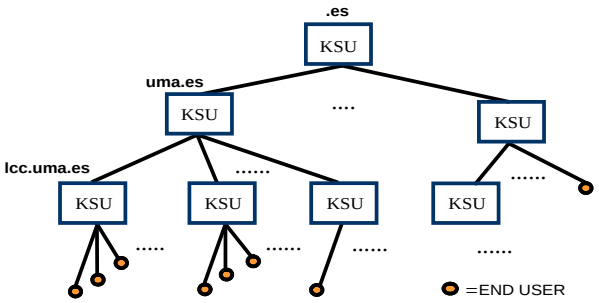

Fig. 2 Hierarchy of Cert'eM Nodes

Operation of the CS can be adjusted according to the characteristics of the node where it is located, and it is the task of the CK to manage the CS configuration. In our implementation, the service provided by the CS, for both Unix and MS Windows platforms, accepts requests through the port 850. In Unix platforms, the server can be installed either as an *inetd* service process or as an independent process

- *Local certificates database.* The first of the databases that form part of the KSU is the local certificates database. It stores the digital certificates of all the users that belong to the domain managed by the CA of the node. As shown in figure 3, the CK generates those certificates from: (a) the information that the user provides, and (b) other information that the CK creates specifically for every certificate to issue. Information from (a) and (b) is used to generate a standard X.509 digital certificate [16].
- *Recently used certificates database.* The second of the databases is used to store the certificates of those users that are external to the domain. To be more precise, these are the out-of-domain certificates requested by local users when they need to perform either encryption or authentication of documents procedures. In fact, this database operates as a certificate cache, and prevents the CS from performing an excessive number of external requests. The certificate cache, carefully designed, enhances the efficiency of the system without introducing any security risk.

Certainly, any CA can define its own cache policy according to its users needs.
The size of this database can be configured and, as stated, it is filled with external certificates requested by local users. A *Least Recently Used* (LRU) policy is used to extract certificates from it and leave free space for certificates that are requested later. Therefore, whenever a local user requests an external certificate the CS checks if such certificate is in the cache space. If that is the case, the CS does not request the corresponding external KSU the certificate, but a simple confirmation of its actual validity. Thus, we obtain a lower traffic rate among KSUs in the certification hierarchy.
It is important to point out that the certificate of a user is only stored in the local database of his/her KSU and, as shown in figure 3, only the corresponding CA is entitled to place it there. This represents an important advantage over other schemes because the revocation process is facilitated in the case that the private key of a user is lost or compromised. The reason is that revocation operation remains local and does not affect the rest of the system; that is, the CA deletes the certificate from the local database and inserts the new one.
Moreover, in the special case that the power to revoke belongs to the CA of a domain we can avoid most of the problems that other schemes introduce. In other schemes, it is necessary to: (i) revoke all the certificates previously issued by that CA, (ii) issue a new certificate for every user associated with the CA, (iii) send them to the repository, and (iv) notify all users of the incident because their certificates will be invalid during certain period. In Cert'eM, and because of the local management of certificates, the change of a CA's key is transparent to users.
From this explanation we can argue that another of the basic features of Cert'eM and, as previously mentioned, one of its priority goals, is to avoid the use of CRLs. Frequently, systems that use CRLs or similar mechanisms, such as the *On-line Certificate Status Protocol* (OCSP) [17], or the *Suicidal Bureau* (SB) [18], provide means to minimize the number of accesses needed to verify a certificate. However, these solutions are too much artificial and inefficient. Currently, OCSP is the most important online mechanism to obtain the status of a certificate.

Although its name may indicate that OCSP has similarities with our proposal, we must highlight several important differences:

- OCSP is designed as a complement to CRLs, while Cert'eM is an alternative based on a different idea.
- In the OCSP model the certificate status providers, known as *responders*, may not be the CAs, and, if not, these entities must contact the corresponding CA anyway for a response to be useful.

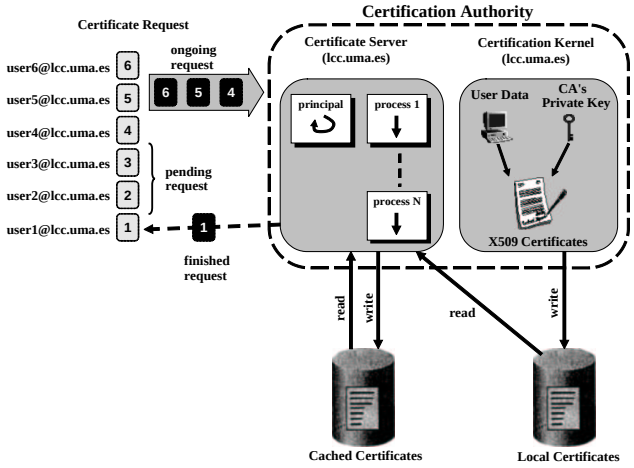

Fig. 3 KSU Components

- Due to the proposed OCSP architecture, a wide range of situations can arise. This in turn results in a high complexity of the system and the protocol, introduces added problems and does not provide any advantage over real online systems such as Cert'eM, which are extremely simple and more efficient.

Another original feature of Cert'eM is the use of validity statements whenever it is necessary to validate a certificate. This concept, already introduced in section 2.2, is a timestamp statement signed by the CA. This one attests that the certificate has not been revoked at the time of issuance of the VS. Other models in the literature propose a similar mechanism. For instance, SPKI/SDSI introduces the concept of *One-time revalidation* (OTR) to indicate that the certificate is valid for a transaction. An OTR server receives a random number and sends, digitally signed, the proof of validity together with that number. The advantage of an OTR is that it does not require that clocks of the systems are synchronized. However, this solution is not convenient for our purposes because it does not provide means to guarantee that the proof of validity is destroyed and not used again later.

Although Cert'eM makes no use of CRLs, this does not mean that CAs do not register the information about the users' certificates that have been revoked. To the contrary, every CA manages locally its own *Local Invalidation Log* (LIL). Notice that a LIL is completely different to a CRL because only the corresponding CA will have access to the LIL in that Cert'eM node.

Therefore, when a user's certificate needs to be invalidated (because his/her key has been lost or compromised, or because the CA has reasons to cease certifying the user) the CA simply deletes the certificate from its local database and stores the revoked certificate in a LIL. This procedure is simple, immediate, requires no communication and can provide proofs of the certificate revocation in case the CA needs those proofs.

Regarding the storage of certificates, it is necessary to point out that a CA's certificate (the certificate of, for instance, *CA@lcc.uma.es* in the figure) is never stored in the local certificate database; it is stored in the database of the upper KSU, that is, the father KSU (*uma.es*, in the figure). The reason is that *CA@lcc.uma.es* is considered as just another user in the *uma.es* domain. Thus, the CA in this domain, *CA@uma.es*, issues the certificate of *CA@lcc.uma.es*. This is a general rule for all the system except for the Root Authority located at *.es*, that is certified by the international authority of domains register, I.A.N.A. or Internic.

It is also important to clarify that a CS never stores certificates in the local database of a Cert'eM node. Its operation is limited to read, under external requests, certificates that have been previously stored by the CK of the node. Each CA can set restrictions to limit the users or KSUs allowed to access the server. This feature provides the CA with a flexible and useful tool to avoid abuse, and to balance the workload between different nodes.

Finally, the general structure of Cert'eM can be considered as an original contribution too. It is possible to find some similarities between this one and the PKI model of the proposal *Secure-DNS* (Secure Domain Name Server) [19]. In both cases, the Internet domain name hierarchy is used to find the location where a particular key (certificate) is stored. However, Secure-DNS uses the *Name Server* files while Cert'eM uses the e-mail offices. Our choice is based on the following reasons:

- Unlike e-mail offices, it is usual that several domains share the same DNS. Therefore, frequently DNSs are not closely related to users and their CAs may not have direct knowledge of a user's identity, with more vulnerability to impersonation.
- DNSs are intended to store information about domains, not about users. As a consequence, they provide a registration procedure for a new domain but not for new users of one of the registered domains. In fact, there is no need that a final user ever interacts with the DNS to get access to Internet. On the contrary, users are forced to interact with e-mail offices to set up an e-mail account.
- DNSs use caching and lifetime mechanisms that could produce inaccurate or false information in some situations. This feature can be used to attack the system.
- One of the main goals of Cert'eM design has been to explore the locations that are more adequate to establish CAs. The criteria being followed has been to use a CA, and by extension, a KSU, in every *source-of-trust*. This is not the criteria of Secure-DNS because in this model CAs are established in the same location as name servers.

3.3 System Operation

In this section we describe the sequence of actions that are carried out when any user (requester) wants to get the public-key certificate of another user (addressee). We must note that the structure of data transmitted by a KSU in response to a certificate request has two components: (i) a X.509 certificate containing, among other information, a serial number and the expected life of the certificate (the validity information); and (ii) the VS signed by the CA, containing the certificate serial number and the time of the VS issuance.

The process to obtain a certificate starts when the requester provides the e-mail address of the addressee to his/her KSU (origin KSU), and this one, in turn, conducts the request to the addressee KSU (destination KSU), whose database contains the certificate. Such operation is simple to do because the system can determine, from the email address provided by the requester, the addressee KSU to be contacted.

The general description of these actions is shown in figure 4 with a more specific scenario. In this case, the figure depicts the information flow produced when user Bob <*bob@r.s.t*> requests the key of user Alice <*alice@x.y.z*>:

- Step 1: Bob provides to his KSU the e-mail address of Alice (*alice@x.y.z*).
- Step 2: Exploring that e-mail address, *Bob's* KSU infers which is the intended KSU where to request for the certificate. In this case, *Alice's* KSU the one at node *x.y.z*.
- Step 3: *Alice's* KSU sends to *Bob's* KSU the reply, containing the certificate.
- Step 4: *B* obtains Alice's certificate, issued by *CA@x.y.z*.

If considered necessary, Bob can also request the certificate of *CA@x.y.z* from the KSU located at *y.z*, obtaining a new certificate to verify the CA signature over Alice's certificate. This is depicted in figure 5. The ascending validation process can continue until a top-level node is reached. In case that no KSU was present at, say, *y.z* (because, for example, the domain does not support Cert'eM system yet), the key of *CA@x.y.z* is automatically requested from the parent node, that is, *z*. This allows Cert'eM to be used even in case of incomplete structures.

The on-line features of CAs in Cert'eM may open a discussion about the security of the system. Traditionally, CAs have been conceived as off-line elements because of the high importance of the operations they perform and because of the inherent risks of attacks in open networks.

Certainly, from the security point of view, on-line CAs may bring some risks, so we have considered some countermeasures. Regarding the access of malicious agents to CAs, we are performing some tests to allocate

CAs functionality in cryptocards within the same computer system. This will provide not only higher speed for certification operations but also an intrinsic security measure from external attacks.

On the other hand, *denial-of-service* (DOS) attacks are an important source of problems too. These attacks are usually carried out by a coalition of multiple computers. In this case the attacks are known as distributed *denial-of-service* (DOS) attacks. The attacks are based on sending a massive number of requests to a server, in order to make it impossible for it to attend to normal requests.

In the case of certification infrastructures, such as Cert'eM, it is essential to provide means to protect the system against these attacks. The solution adopted in Cert'eM is based in the way requests are done. Recall from this subsection that users make requests only to their corresponding KSUs. Likewise, KSUs only accept requests from their local users and subordinate KSUs. This scheme provides robust protection against DOS attacks because only a restricted group of users can send requests to a certain KSU, thus making DOS attacks much more difficult.

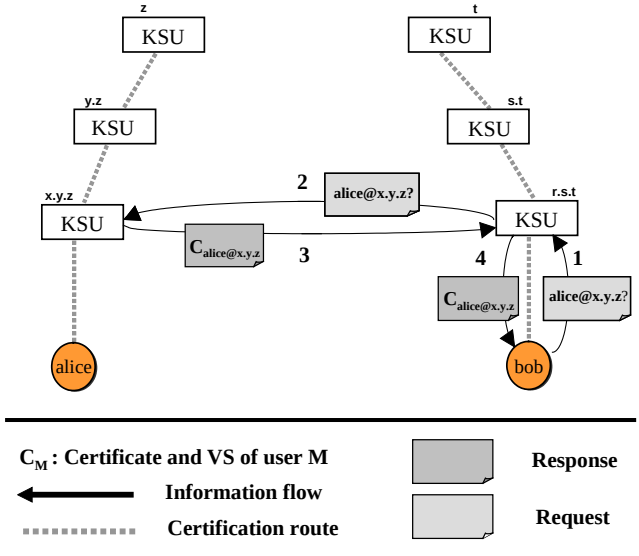

Fig. 4 Example of Information Flow for a Certificate Request

4 Access to the Certificate Server: Description of the Protocol

In this section we introduce the protocol that describes how individual users and other key servers access a KSU. The requests are represented in a Client/Server scenario. Note that either individual users or KSUs can play the client role. For instance, a request from user *bob@r.s.t* (client) to his KSU, located at *r.s.t* (server), may be

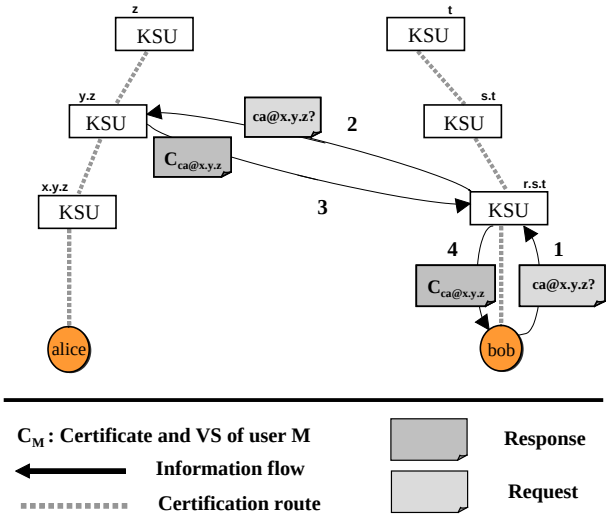

Fig. 5 Information Flow for Certificate Verification

followed by a request from this KSU (now client) to the KSU located at *x.y.z* (server). In the subsequent description *C* will be used to denote a generic client, while *S* will denote a generic server.

4.1 Protocol Data

We will use the following data structures as part of the protocol:

<*clientID*>: Identification of the client.

<*userID*>: The e-mail address (name@domain) of the user whose key (certificate) is requested. Cert'eM uses the domain to determine in which KSU the key resides.

<*cert*>: An X.509 certificate containing among other information: the user's identification (equivalent to <userID>), the user's public key, a certificate serial number that is unique, and the expected activity period life of the certificate. This record is kept in the KSU database, so there is no need to produce it online.

<*vs*>: A timestamp statement containing a certificate serial number, and the time of issuance of this <vs>, signed by the corresponding CA. It is used to guarantee that the certificate with that serial number was not revoked at the time of issuance. Unlike the <cert>, this record is produced online.

<*certID*>: Certificate identification consisting on the <userID> of the addressee user and the certificate serial number of the active certificate to be checked.

4.2 Protocol Description

The protocol is structured in three phases: connection, transaction and termination, which are described next, and represented in figure 6.

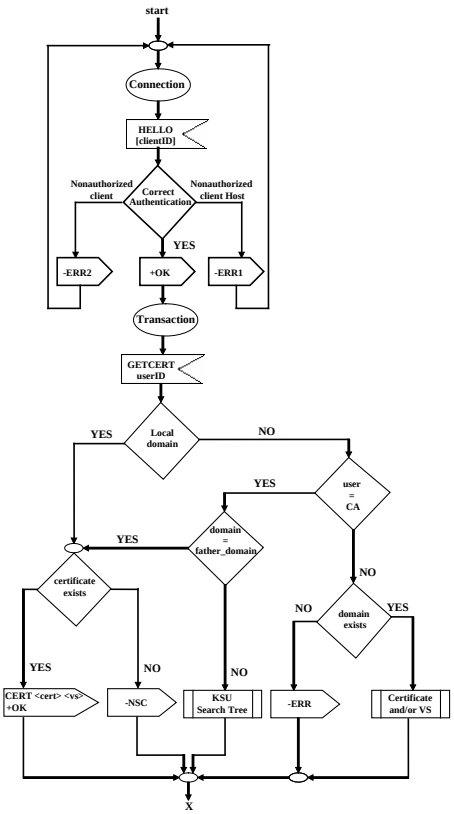

Fig. 6 Certificate Allocation

4.2.1 Connection Phase The connection is established with the following message:

C : HELLO [<clientID>]

where <clientID> is optional, depending on the particular KSU security policy to be implemented. Each CA can set restrictions to limit the users or computers allowed to access the server. When a server receives this message, it checks whether <clientID> is allowed to establish the connection. Afterwards, the server sends one of the following messages as a response:

S : +OK – the client has permission
S : -ERR1 – the client host is not allowed
S : -ERR2 – the client <clientID> is not allowed

4.2.2 Transaction Phase When the connection is successfully established the client can start requesting keys. For this purpose the following message is used:

C: GETCERT <userID>

The following situations can arise when the server receives the previous message:

1. The *domain* requested matches the *domain* of S (i.e. the requested key belongs to a local user of S). The response is:

   S : CERT <cert> <vs>
   S : +OK

   if the certificate was found, or:

   S : -NSC –no such certificate

   if not found.
   In some cases, the addressee user may have more than one certificate, each one for a different purpose. This means that the *key usage* extension field of each of the certificates will indicate the purpose for which the key is used: digital signature, non-repudiation, key encryption, etc. In our scheme, the requester gets all the certificates of the addressee. The requester will choose, by analyzing the *key usage* extension field of the certificates received, the one he/she needs. Note that the amount of possible certificates for one user is very low (normally one) in most cases, so this way of operating does not introduce any performance inconvenience. Accordingly, all steps of the protocol that contain a "*CERT <cert> <vs>*" response are just a simplification of the general procedure. That is, one of these messages is sent for every certificate of the addressee user, in case that the user has more than one.
2. The requested *domain* does not correspond with that of S:
   (a) The requested *name* is CA:
      i. If the *domain* of S corresponds to the parent of the requested *domain*, then the certificate may reside in the database of S; therefore, the case is managed as a local certificate request (step 1).
      ii. Otherwise, the certificate is requested from the KSU located at the upper node of *domain*. If there is no KSU in that node the request is redirected to the succeeding upper one until the top-level node or S are reached.
   (b) The requested *name* is not CA:
      i. If *domain* does not exist the server returns an error message:

         S: -ERR3

      ii. Otherwise, a new connection is established to request the certificate from the KSU located at *domain*. The result of this new request is forwarded to the requester.

Step 2.a.ii makes reference to the establishment of a new connection to request for an external certificate, which corresponds with the section "Certificate and/or VS" of figure 6. From here, two alternatives arise (figure 7).

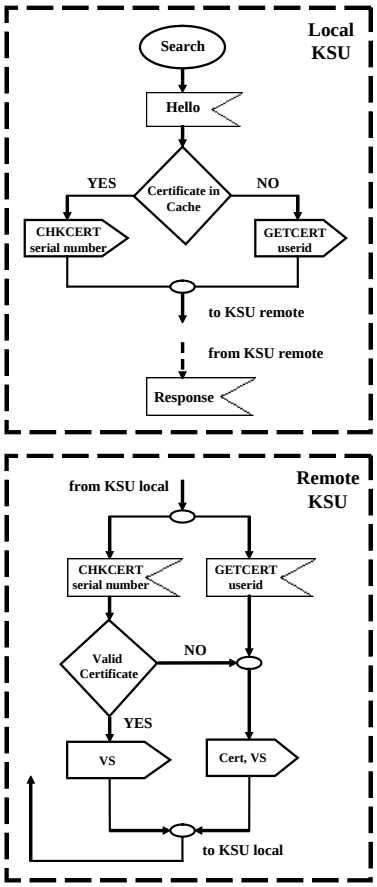

Fig. 7 External Access

The first possibility is that the requested certificate is not in the cache. In this case, the KSU performs a usual certificate request to the remote KSU:

C: GETCERT <userID>

The response of the remote KSU will be:

S : CERT <cert> <vs>
S : +OK

if the certificate was found, or:

S : -NSC

if not found.

As we can see, the KSU operates like a client in this case, and passes the request received from the user to the remote KSU. This operation is the same as in step number 1.

The second possibility is that the requested certificate is in the cache. If the KSU has the certificate of *userID* (issued in t_n), then there is no need to request for all the information of the certificate. It is only necessary to verify that it has not been revoked, or what is the same, that no certificate for that user exists after t_n. In this case, the local KSU (behaving as a client), sends to the remote KSU (behaving as a server), the following message of certificate verification:

C: CHKCERT <certID>

And the remote KSU response:

S: VS <vs>
S: +OK

in the case where the certificate is still valid (has not been revoked). This is known by simply verifying that the certificate corresponding to the serial number included in <*certID*> is still in the local certificates database.

In the other case, the remote KSU sends the new certificate:

S : CERT <cert><vs>
S : +OK

In both cases, the answer can be:

C: -NSC

if the certificate was not found.

4.2.3 Termination Phase This phase is to inform the server that the client has finished the request of certificates. The client sends the message:

C: EXIT

And the server response:

S : +OK

5 Important Implementation/Deployment Details

Although in previous sections we have given some details about the implementation of our system, in this section we provide a more general information concerning not only implementation, but some important issues to consider when deploying Cert'eM. Note that we discuss only those aspects that we believe are more relevant attending to the features that are original contributions in Cert'eM.

The details correspond to the Cert'eM prototype that has been tested in an electronic forms signature application for the Transport Council of the regional government [20], and is actually under evaluation by RedIRIS, the National Research & Academic Network in Spain[1].

[1] http://www.rediris.es/cert/proyectos/certem/test.html

In a more local scenario, the same prototype is being used by a wide group of selected students at the University of Malaga. Our scheme is used as the authentication infrastructure necessary for running certified electronic mail applications.

5.1 Determining Physical Location of KSUs

We have shown that in the second step of figure 4 (also in figure 5) the origin KSU requests the certificate from the destination KSU by exploring Alice's e-mail address. As we have seen, this process is simple from a logical point of view. However, the task of learning the specific physical address of the target KSU is quite elaborated, and essential for Cert'eM operation. We detail in this subsection how this procedure is done.

The origin KSU learns where to point the certificate request that its local user needs by using the information contained in the DNS. This is an essential query element in our KSU location scheme. The sequence of steps is represented in figure 8 for the case in which a user of another domain requests from its KSU the certificate of <*jlopez@lcc.uma.es*>:

1. Once the user of the other domain requests from his/her KSU the certificate of <*jlopez@lcc.uma.es*>, the KSU tries to resolve the IP address of the domain <*lcc.uma.es*>. An entry in the DNS should exist with the following format (where *111.111.222.222* is just an example of IP address):

   lcc.uma.es IN A 111.111.222.222

2. If no entry of that type is present, the KSU makes a second request to the DNS. This request is related to the intrinsic nature of Cert'eM. More precisely, because the certificate identifier is the user's e-mail address, Cert'eM pretends to establish a link between the e-mail office in the domain and the CS of that domain. Therefore, the second request to DNS is related to MX - Mail Exchanger - register of the e-mail office [21]; that is, the KSU looks for a DNS entry like the following ones:

   lcc.uma.es MX 5 111.111.222.222

   We must underline that in the case that more than one MX register exists for a specific domain, the content of the preference field (5 in the example above) will indicate the entry to choose.
3. Finally, if the second step fails to find the MX record, the KSU makes a request to the DNS using the name domain prefixed with "certem-tcp". For example:

   certem-tcp.lcc.uma.es IN A 111.111.222.222

   This alternative forces the DNS manager to implicitly find that entry. This is convenient, for instance, when the mail traffic supported by the e-mail office is high. In this case, the e-mail office and the KSU should not coexist in the same computer in order to avoid a low response time (and low efficiency) of the KSU.
   An additional reason that has made us to develop this additional search mechanism is the possibility of creating virtual KSUs. That is, we open the possibility to install different KSUs in the same computer in such a way that domains with low number of users can be allocated together. This is achieved by setting the domains involved to the same physical address:

   certem-tcp.lcc.uma.es IN A 111.111.222.222
   certem-tcp.crypto.lcc.uma.es IN A 111.111.222.222

5.2 External LDAP Connections

The use of information directories, such as LDAP [22], is a common practice nowadays. In fact, and regarding PKIs, directories are one of the most widely deployed mechanisms for distribution of information concerning certificates and CRLs.

In order to achieve an appropriate interaction of LDAP users with a Cert'eM system, we have implemented a transparent proxy service that allows access to a Cert'eM service through the port 389 of a LDAP-Cert'eM gateway.

In this way, we establish our *ldap-proxy* service in the port 389 of the gateway, waiting for incoming requests from LDAP clients. In case that client requests for information that is not related to certificates (e.g., a telephone number, an e-mail address, etc.), the request is redirected to the original LDAP server, that is reallocated in the port 3890. The information requested is transparently delivered to the LDAP client. On the other hand, if the port 389 receives a certificate request, the gateway takes the role of client, connects to the appropriate KSU, and obtains the certificate. The certificate is then delivered to the LDAP client.

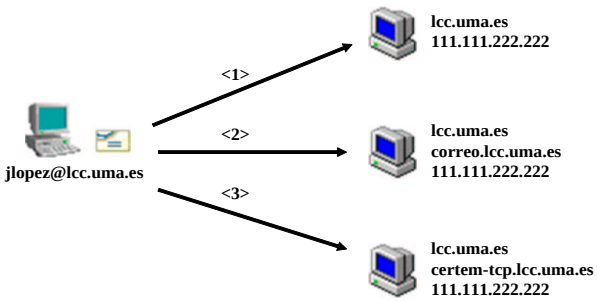

Fig. 8 Algorithm for KSU location

6 Conclusions

Problems associated with digital certificates management, like storage, retrieval, maintenance, and specially, revocation, require special procedures that ensure reliable features because of the critical significance of inaccuracies. In this paper, we have presented a scheme, Cert'eM, a key management and certification system based on the structure of the electronic mail service and on the principle of near-certification.

Cert'eM provides secure means to identify users and distribute their certificates, eliminating problems associated with common revocation procedures, as well as simplifying the validation of certificates.

We have considered the revocation problem as priority in the design process because it has a big influence on the rest of the PKI components. Therefore, we have developed an alternative solution to the use of CRLs, a type of mechanism that we consider inefficient for many Internet applications. In fact, we believe that the concept of CRL itself is a drawback, and that a better solution is one where the knowledge of a revoked certificate is immediately available to users, if this brings no degradation of performance in the system.

We have shown in the paper how our system defines a hierarchy of KSUs, essential components of Cert'eM and that integrate both key certification and certificate management functions. As explained, KSUs operate over disjoint groups of users, conforming a hierarchy that is parallel to the hierarchy of Internet domains in such a way that KSUs are associated with Internet e-mail offices.

We have also shown the composition of the KSUs, how the components operate inside the whole system, and the way in which KSUs are located (IP addresses inferred with the support of DNSs) inside the set of KSUs forming the PKI.

Finally, we have examined in detail the protocol for accessing Certificate Servers, the elements of KSUs that receive certificate requests and deliver the certificates to those requests.

References

1. W. Ford, M. Baum, "Secure Electronic Commerce: Building the Infrastructure for Digital Signatures and Encryption "(2nd Edition), Prentice-Hall, 2000
2. J. Kohl, "The Use of Encryption in Kerberos for Network Authentication", Advances in Cryptology - CRYPTO'89, Springer, 1989, pp. 35-43.
3. J. Kohl, B.C. Neuman, "The Kerberos Network Authentication Service (V5)", Internet Request for Comment 1510, September 1993.
4. D. Davis, "Kerberos Plus RSA for World Wide Web Security", First USENIX Workshop on Electronic Commerce, 1995, pp. 185-188.
5. R. Ganesan, "Yaksha: Augmenting Kerberos with Public Key Cryptography", Internet Society Symposium on Network and Distributed Systems Security, 1995, pp. 132-143.
6. J. Schiller, D. Atkins, "Scaling the Web of Trust: Combining Kerberos and PGP to Provide Large Scale Authentication", USENIX Technical Conference, 1995.
7. W. Diffie, M. Hellman, "New Directions in Cryptography". IEEE Transactions on Information Theory, IT-22, n 6. 1976, pp. 644-654.
8. R. Clarke, "Human Identification in Information Systems: Management Challenges and Public Policy Issues", Information Technology & People, v. 7, n. 4. pp. 6-37, 1997.
9. S. Garfinkel, E. Spafford, "Web Security and Commerce, 2nd Edition", O'Reilly & Associates, 2001.
10. European Commission, "Directive 1999/93 of the European Parliament and the Council on a Community Framework for Electronic Signatures", December 1999, Official Journal L 013, 19/01/2000 pp. 0012 - 0020.
11. B. Wright, "Making Numbers Ceremonial: Signing Tax Returns with Personal Identification Numbers", personal communication, 1998.
12. L. Detweiler, "Identity, Privacy and Anonymity on the Internet",http://www.rewi.huberlin.de/jura/proj/dsi/Netze/privint.html.
13. ISO International Standard 9594, "Information Technology - Open Systems Interconnection Reference Model: The Directory", 1988.
14. International Telecommunication Union, ITU-T Recommendation X.509, "Information technology - Open Systems Interconnection - The Directory: Authentication Framework", 1997.
15. J. Lopez, A. Maña, J. J. Ortega, J. M. Troya, "Distributed Storage and Revocation in Digital Certificate Databases", 11th International Conference on Database and Expert Systems Applications (DEXA'00), pp.930-938, LNCS 1873, Springer Verlag, September 2000.
16. International Telecommunication Union, ITU-T Recommendation X.509, "Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks", March 2000.
17. M. Myers, R. Ankney, A. Malpani, S. Galperin, C. Adams, "X.509 Internet Public Key Infrastructure. Online Certificate Status Protocol - OCSP", Internet Request for Comment 2560, 1999.
18. R. Rivest, "Can we Eliminate Revocation Lists?", Financial Cryptography, 1998, Springer 1998, pp. 178-183.
19. D. Eastlake, "Domain Name System Security Extensions", Internet Request for Comment 2535, March 1999.
20. J. Lopez, A. Maña, J. A. Montenegro, J. J. Ortega, J. M. Troya, "Designing Software Tools for the Use of Secure Electronic Forms", 3rd ACIS Int. Conf. on Software Engineering, Artificial Intelligence Networking and Parallel/Distributed Computing (SNPD'02), pp.157-163, June 2002.
21. P. Mockapetris, "Domain Names - Concepts and Facilities", Internet Request for Comments 1034, November 1987.
22. J. Hodges, R. Morgan, "Lightweight Directory Access Protocol (v3): Technical Specification", Internet Request for Comment 3377, September 2002.