The Computer Journal Advance Access published May 6, 2013

doi:10.1093/comjnl/bxt037

Certificate Revocation List Distribution System for the KAD Network

JUAN CAUBET*, CARLOS GAÑÁN, OSCAR ESPARZA, JOSE L. MUÑOZ,
JORGE MATA-DÍAZ AND JUANJO ALINS

Department of Telematics Engineering (ENTEL), Universitat Politècnica de Catalunya (UPC), Barcelona, Spain
**Corresponding author: juan.caubet@entel.upc.edu*

Many peer-to-peer (p2p) overlays require certain security services which could be provided through a Public Key Infrastructure. However, these infrastructures are bound up with a revocation system, such as Certificate Revocation Lists (CRLs). A system with a client/server structure, where a Certificate Authority plays a role of a central server, is prone to suffer from common problems of a single point of failure. If only one Authority has to distribute the whole CRL to all users, perhaps several millions in a structured p2p overlay, a bottleneck problem appears. Moreover, in these networks, users often have a set of pseudonyms that are bound to a certificate, which gives rise to two additional issues: issuing the CRL and assuring its freshness. On the one hand, the list size grows exponentially with the number of network users. On the other hand, these lists must be updated more frequently; otherwise the revocation data will not be fresh enough. To solve these problems, we propose a new distributed revocation system for the Kademlia network. Our system distributes CRLs using the overlay itself and, to not compromise the storage of nodes, lists are divided into segments. This mechanism improves the accessibility, increases the availability and guarantees the freshness of the revocation data.

Keywords: Certificate Revocation List (CRL); structured peer-to-peer (P2P) overlay; KAD network

Received 5 August 2012; revised 20 February 2013
Handling editor: Jongsung Kim

1. INTRODUCTION

Peer-to-Peer (p2p) systems emerged as an incipient paradigm of communications, which allow sharing resources without the need of centralized servers. Nowadays, p2p networks are having much success due to the availability of cheap bandwidth and the growing number of computers sharing services and resources. However, the most popular p2p overlays have security problems due to both the lack of a central authority and the assumption that nodes behave honestly [1]. For these reasons, and others such as privacy [2, 3], these networks are hardly being used for commercial applications.

In this context, deploying a Public Key Infrastructure (PKI) within the overlay will allow peers to communicate and share information securely. Thus, they will be able to perform transactions, like electronic payments, or even to sign digital contracts without risking their confidential data.

PKI was developed to meet with the main security requirements in the Internet environment, adopting the idea of public key and employing the digital certificate concept to bind the entity information and relative public key. But using digital certificates implies the need to validate them [4]. When a user wants to access a resource, the provider needs not only to find a certificate chain from provider to user, but also to check that all the certificates in the trust chain are valid. However, the provider also needs to be able to remove misbehaving and malicious peers. Thus, digital certificates can be revoked due to several reasons. For example, a certificate will be revoked if the private key related to the certificate has been compromised or the affiliation of the owner has changed and therefore the certificate is no longer valid. And so far, several revocation systems have been proposed in the literature. The traditional approach for providing revocation information is based on Certificate Revocation Lists (CRLs) [4].

CRLs are blacklists issued periodically by the Certificate Authorities (CAs) that enumerate revoked certificates along with the revocation date and, optionally, the revocation reasons.

Other traditional mechanisms are based on querying an online server able to verify the revocation status of a certificate, e.g. the Online Certificate Status Protocol [5]. These revocation systems, and other less popular ones, have different features in terms of network traffic overhead, load on the servers that provide revocation information, freshness of this information and suitability for offline usage. But most of these are typically client/server structures, where a CA plays a role of central server, and suffer from the common problems of a single point of failure. In addition, if only a few CAs (or just one) distribute the entire CRL to all network users, a bottleneck problem appears, even using a two-level caching protocol [6].

In the overlay context, if only few CAs are capable of distributing the CRL to all peers, they will become overloaded, as a structured p2p overlay can have hundreds of thousands of users, or even millions. And if we take into account that users will own a set of pseudonyms, each one with an associated certificate, the problem is accentuated; as the CRL size[1] grows exponentially with the number of network users. In addition, if the network has a large number of users and each user has a set of certificates, the certificate revocation rate is expected to be very high. Therefore, CRLs must be updated more frequently; otherwise the checked data by the users will not be fresh. Thus, typical CRL updating policies are not enough to ensure the availability and the freshness of the revocation data.

On the other hand, there are some people in the research community who are unwilling to accept the use of any kind of centralized service in a structured p2p overlay, as these networks are supposed to be pure p2p networks.[2] Therefore, in this type of networks, it is not enough to replicate the CRL originator server to solve the bottleneck problem.

To solve aforementioned issues, we propose a new distributed revocation system for a particular p2p overlay, the Kademlia network (KAD). We distribute CRLs using the overlay itself, and to not compromise the storage of peers, the CRLs are divided into segments. By storing several replicas of `keywordIDs` and `sourceIDs` in different peers, our mechanism highly improves the accessibility of the CRL segments, avoiding bottleneck problems.

With our mechanism, the CA is just another peer of the network responsible for publishing and storing all new CRL segments in the overlay. However, unlike in traditional methods, the normal peers do not only download CRL segments from the CA or other peers, but in turn they also provide such segments to other peers. Thus, the number of owners of CRL segments that can provide the revocation data increases exponentially, which certainly increases the availability. Moreover, this novel way of distributing CRL segments allows us to improve the CRL updating policy maintaining all the security properties of the traditional system. As the CRL segments can be updated separately, the freshness of the revocation data can be highly improved at a lower bandwidth cost.

The rest of the article is organized as follows: in the next section a brief related work is given. Section 3 presents the background of the KAD network. Section 4 describes the operation of our system, and its performance analysis is given in Section 5. And finally, we conclude in Section 6.

2. RELATED WORK

Ying and Jiang, in [8], propose a new certificate revocation system based on Chord [9], where they use a bloom filter to resolve the bottleneck problem of some peers. Their method consists in flooding all nodes with the bloom filter vector of a certain CRL segment. In this way, they save storage and bandwidth but the false positives that occur in a bloom filter complicate the certificate validation process. Furthermore, using flooding mechanisms may be inefficient for networks with hundreds of thousands of users.

In [10], the authors propose a *Super Peer* method to enhance the p2p architecture introducing hierarchy in the distributed structure. Peers act as both servers and clients super peers act as authorities to clients, and CRL distribution is carried out through a pull mechanism. But this hierarchy does not prevent storage problems in peers nor the great bandwidth consumption if the CRLs are very large.

Morogan and Mutfic describe, in [11], a distributed system for certificate revocation based on the distribution of CRLs by p2p networks. They aimed to achieve good offline functionality and to improve the CRL server's availability using Delta CRLs. However, in networks with many revocations, these updates may have considerable size and do not avoid problems that arise when the CA issues a new CRL. In addition, the authors do not propose a system one hundred percent distributed.

In [12], the authors propose a new distributed trust infrastructure based on the Chord network and within this infrastructure describe a revocation system. They use the nodes to store the revocation information, but without using CRLs. Each node is responsible of storing a set of certificates, and if a certificate is revoked, the same node stores a tag that indicates the revocation.

On the other hand, *trust* is another approach for managing access control in P2P networks. In these networks, trust captures the liability, trustworthiness and satisfaction of a peer. The main goals of trust models are to increment the percentage of successful exchanges and to ensure the models' scalability and simplicity. For instance, the BBK model [13] described a quantified trust, which divides trust into two types: direct trust and reference trust. However, in this model the success and failure of trust are scaled the same, which may result in malicious references. In [14], the authors presented a distributed

[1] Note that in this article we talk about the size of an entire CRL, or segment, to refer to the number of certificates that are stored in that list.

[2] Pure decentralized networks use a p2p scheme in all their processes and there is no central server at all. A typical example of such networks is given by Gnutella [7].

trust model for the JXTA platform. In this model, trust is computed based on users' interests and keywords. Nevertheless, a configuration table is needed for the whole group, which may result in inefficiency in a dense P2P network. In [15], the authors described different types of trust and defined the relation between trust values and roles in a access control model. However, this model lacks in determining how the role based on an access control model works well in a trusty network.

3. BACKGROUND ON KAD

KAD is the widest deployed p2p overlay in the Internet with an estimated number of concurrent online users of around 4 million. It is based on the Kademlia Distributed Hash Table (DHT) routing protocol [16] and is implemented by eMule [17] and aMule [18] file sharing applications, both open source. Each node in KAD has a 128 bits `KADID` which defines its position in the virtual space of the overlay, and the distance between any two points (nodes or contents) is defined as their bitwise exclusive-or (XOR) interpreted as an integer. Thus, KAD treats nodes as leaves in a binary tree, in which their position is determined by the shortest unique prefix of the `KADIDs`. Nodes group their tree contacts in buckets and store these lists as routing tables. Each node registers a maximum of k contacts per level i, contacts that are at a distance of between 2^{128-i} and 2^{127-i} from the `KADID` of the node regarding the XOR metric. The deeper the contact is in the tree, the closer it is to the node and the better the node knows this part of the DHT; this provides routing in $O(\log n)$. Routing to a `KADID` is done in an iterative way; messages are forwarded to the n closest contacts to the target and each node on the path to the target `KADID` returns the next hop.

As in many other overlays, the purpose of the DHT is to bind files and keywords. To share a file, raw data and keywords must be hashed separately using the MD5 function and then published in a distributed way several times (at least 10 times). KAD only publishes references, metadata and sources; and these references are stored in nodes which are close enough in the overlay to the `keywordIDs` and `sourceIDs`, respectively. This distance is called the *tolerance zone* of a `KADID`, and is calculated using the eight most significant bits of the identifier. Moreover, to improve availability, resources are periodically republished: `sourceIDs` every 5 h and `keywordIDs` every 24 h. Analogously, a node, on which a `sourceID` or `keywordID` was published, will delete the information after 5 and 24 h, respectively, if it is active. Republishing is done exactly the same way as publishing.

4. CRL DISTRIBUTION SYSTEM

In a p2p overlay with hundreds of thousands of users, or millions, if each user has a set of pseudonyms, the number of certificates issued by the CA can be up to hundreds of millions. For this reason, two problems arise that are not taken into account in the traditional revocation systems. On the one hand, the size of the CRLs is quite large, which hinders their distribution. On the other hand, it appears a trade-off between the freshness of the CRL and the overhead caused by its distribution. Therefore, it is necessary to define another type of CRL distribution system that optimizes the availability of the revocation data and its freshness.

4.1. System requirements

(1) *Scalability:* The cost of validating the status of certificates must be low enough. This is due to the fact that p2p overlays may be very large networks, involving from tens of CAs to millions of users conducting billions of transactions.
(2) *Load balancing:* The cost of storing and distributing CRLs must be distributed among all the peers.
(3) *Tolerance to frequent changes:* The system must tolerate frequent joins and leaves, as p2p networks are characterized by the lack of peer stability.
(4) *Performance:* The entire system must be efficient, without requiring excessive computational complexity or network overhead for any participant.
(5) *Unforgeability:* No entity should be capable of generating fake revocation information.

4.2. Overview

Our proposal divides the entire CRLs into several segments, to avoid CRLs from becoming unmanageable due to their size. These CRL segments are stored and shared in a distributed way, improving their availability and preventing the CAs from becoming bottlenecks.

All network nodes are considered as potential servers of CRL segments, and the CAs act like normal nodes within the overlay. However, the CAs are still responsible for issuing the CRLs and inserting the CRL segments into the network. The introduction of the segments in the overlay is performed using a pull mechanism; that is, initially the only server to download a certain CRL segment from is the CA that issued it. Of course, CAs are also responsible for dividing the CRLs into segments and to avoid that a node could generate fake CRL segments, or modify a valid segment. To achieve this degree of security, they sign all the segments using its cryptographic public key in the same way they do with the standard CRLs.

The CRL segments are shared by the nodes from the first time that these are stored. Thus, each time a node downloads a CRL segment, this node becomes a new server within the overlay of that segment. When a CA updates a CRL segment the process starts again, i.e. initially the CA is the only server and at the same time nodes download the segment they become servers of it.

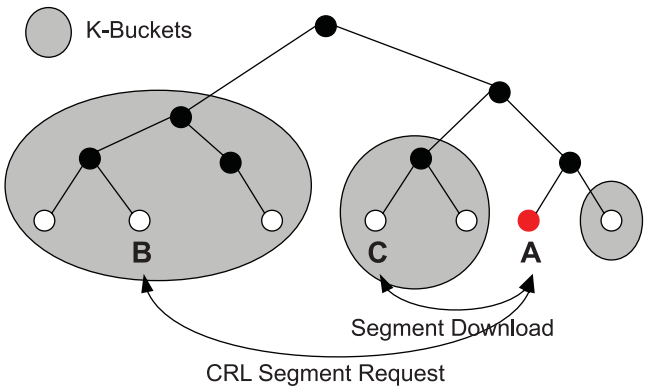

FIGURE 1. Example of the search for a segment in the KAD network.

Taking into account the KAD network features, once a node has a chunk of a segment it begins to share this chunk[3] with the rest of the nodes, although it has not downloaded the entire segment yet. In this way, the CA does not become a bottleneck when the network is initialized or a segment is updated, even if the number of requests in a short period of time is very large, as the overlay itself is responsible for distributing the requests to the new servers.

Next we describe the system performance, from the point of view of a node A, by means of an example, which is shown in Fig. 1. Let us suppose that A wants to verify whether the certificate c is valid. First of all, the node A calculates, by a simple hash function, in which CRL segment should the certificate c be stored if it was revoked. In this case, c should be stored in the CRL segment i. Once A knows which CRL segment needs to verify, it checks if the segment i is stored in its local memory or not. If the segment is stored in the local memory, A checks whether the serial number of c is within the segment. Otherwise, A sends a CRL segment i request to the network. Then, this request reaches node B which in turn responds to A indicating that node C has the CRL segment i. Finally, node A downloads this segment from node C and checks whether the serial number of the certificate c is stored in it. If the serial number of certificate c is stored within the segment, then it is not valid, otherwise c is not revoked.

Note that the KAD network operation is quite more complex than the example shows and the number of nodes which respond to a request is considerably larger, also the number of nodes that have a certain segment is usually much greater than one.

4.3. CRL segmentation

CRLs are divided into 2^k segments according to the serial number of the certificates. To that end, CAs use a hash function, $h(\cdot)$, which given a certificate serial number returns the number of the segment where that certificate should be stored in case of being revoked. In this way, each time the CA revokes a certificate, it calculates the hash of the certificate serial number and stores it in the CRL segment indicated by the output of the hash function. Similarly, nodes also calculate the hash of a certificate serial number if they want to verify the certificate validity, and thus, they obtain the number of the segment that they must consult.

Regarding the implementation of the system, note that the number k is given by the length of the hash function used, as the length of the serial numbers of all certificates may not be the same; each CA can use different lengths.[4] Therefore, each segment will always contain the same potential range of certificate serial numbers independently of the length of the validated certificate serial numbers.

The simplest hash function that meets the above requirements is the modular hash function; $h(c) = c \bmod 2^k$, where c is a certificate serial number. In this way, the maximum size of the segments will be limited and equal for all them. Taking into account that the number of issued certificates can be at most 2^n, each segment will store a maximum of 2^{n-k} revoked certificates. And owing to the simplicity of the hash function used, the computational overhead introduced in the CAs and the nodes is minimum.

To create the different segments, or to calculate a desired segment number, it is not necessary to know the real size of the CRL at the time. CAs and nodes only need to know the maximum size that the entire CRL could have. Therefore, the number of segments in which CAs divide the CRLs is independent of their size. Hence, k must be selected when dimensioning the system. If the CRLs are expected to be really large, k must be large too and vice versa. In this way, the system performance is improved.

Figure 2 shows an example where a CRL, regardless of its size, is divided into 128 CRL segments. For this, the CA calculates the *module 128* of each revoke certificate serial number and obtains the number of the segment where it must store each serial number. Using the same operation, all users obtain the number of the CRL segment where they need to search. Note that, for simplicity, the CRL only contains certificate serial numbers, as our system does not need to use any other information.

Note that in some cases there may be empty CRL segments, but even if a segment does not store any revoked certificate, its publication is also necessary. Nodes must be able to check whether a certificate is valid or not, by means of a document signed by a CA. For security reasons, the fact that a node does not find a certain segment is not enough to assure that a certificate is valid. Therefore, empty signed segments must be published and distributed among the network peers.

[3]Chunks are fragments of information which are downloaded or managed by p2p programs.

[4]As is denoted in [4], the length of the serial numbers of the certificates must be up to 160 bits.

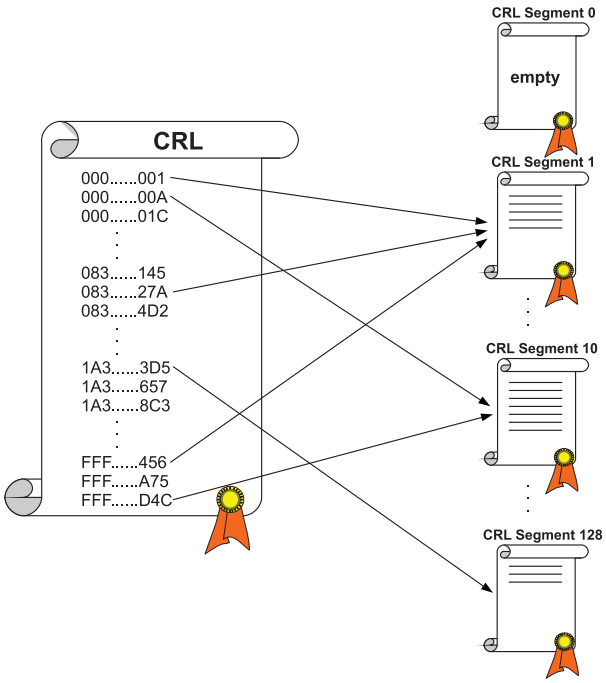

FIGURE 2. CRL segmentation.

4.4. CRL sharing

CRL segments are shared by the KAD network like any other resource. As aforementioned, each CRL segment has a number which is used by the nodes to find the segment that they need to check. The name of the CRL segments within the overlay is of the form '*CRL_CA*$_i$ Segment xxx', where *CA*$_i$ identifies the CA that issued the segments and *xxx* represents the CRL segment number with a length of *m* bits, e.g. *CRL_CA*$_1$ *Segment 10*. On the other hand, pointers or references to segments are stored in several nodes within the overlay with three associated keywords: *CRL*, *Segment* and the segment number (*xxx*).

As we have explained in Section 3, resources are located within the KAD network through their `keywordIDs` and `sourceIDs` (*hash*(*keyword*) and *hash*(*file*), respectively). Therefore, when a node wants to download a new CRL segment, it performs a search for one or more related keywords, which give pointers to multiple sources. Then, these sources return which nodes have the desired segment. Finally, the node starts to download the desired CRL segment.

Once the node already has at least a chunk of the CRL segment stored in its memory, it performs the publication of that segment. To that end, the node publishes two types of references, *metadata* and *sources*, which are sent to nodes in its *tolerance zone*. Keywords are distributed metadata that reference to sources. Sources are the location information pointing directly to the node, which keep a copy of the CRL segment. In KAD, all metadata and sources are replicated in tens of nodes within the overlay, so the searches are performed faster and return more results. It is noteworthy that in the case of the CRL segments, three keywords are used to publish the metadata and the sources. One keyword differentiates the segments issued by the same CA and the other two are used to differentiate the CRL segments from the other resources.

Continuing with the example of the Fig. 2, let us consider that a node needs to verify the validity of a certificate with serial number 'FFF.....D4C'. First of all, it calculates the hash value of this serial number and obtains the value 10. Then, the node searches for the *CRL_CA*$_i$ *Segment 10*. With this search, the node identifies a considerable amount of nodes that have stored this segment. Finally, it downloads the segment from the node that it prefers, and in turn publishes the same CRL segment. Note that initially only the CA has published and stored the CRL segment number 10.

4.5. CRL issuance

As time goes by, CRLs must be updated by the CAs that have issued them. The new revoked certificates must be added in the lists and expired revoked certificates must be removed from them.

In the traditional way of issuing these lists, new CRLs are issued periodically, normally every 24 h. Therefore, every time a new CRL is issued, all nodes that want to verify the validity of a given certificate must download the new CRL. However, updating CRLs' data periodically without taking into account the certificate revocation rate and the network activity may cause problems about the freshness of the revocation data. If the overlay activity is very high and many certificates are revoked each hour, a node can be consulting a CRL issued for 12 h that contains outdated information, or lacks relevant information.

For this reason, in this proposal, CRLs are not issued periodically. They are only issued when CAs have enough new information to update or have not been updated for a considerable time. Moreover, our revocation mechanism takes into account the fact that when a new entire CRL is going to be issued, there are many CRL segments that do not have new information to update. Therefore, it is more efficient to issue each segment separately only when necessary. So, the CRL segments are managed as entire CRLs and issued independently by the CAs, that is, a CA adds and removes certificates from the CRL segments whenever a certificate is revoked or expired, and, when necessary, the CA issues an updated CRL segment.

In this way, we define the variable U_s calculated by the CAs for each CRL segment, which indicates whether the segment must be updated or not. If its value is above a certain threshold, it is not necessary to update the segment, otherwise the CA will issue a new CRL segment. This calculation depends on the number of the network users (N), the certificate revocation rate (R_c) and the number of revoked certificates added to that segment (Δ_s):

$$U_s = \frac{N}{R_c \Delta_s}. \quad (1)$$

Note that the number of revoked certificates in a segment can increase relatively fast due to a high certificate revocation rate. Therefore, the amount of time that the CA does not update a CRL segment becomes shorter, as there is a risk that the new revoked certificates are in use. This is a problem of data freshness. However, although both variables may be closely related, not always a high certificate revocation rate means an increase in the size of a particular CRL segment, and vice versa. For this reason, we have defined U_s.

From the point of view of nodes, there must be a way of knowing whether a CRL segment is fresh or not. And taking into account that all CRL segments are updated at different instants and not periodically, each segment must be time-stamped. To do so, we use two fields of the CRL segment, *This Update* and *Next Update*. The first indicates the time when the segment was issued and the second when this segment will be updated. Of course, this second time defines the maximum amount of time that the segment can remain outdated.

Taking into account that the CRL segments can be updated at any time within that period, the CA responsible for a segment is the only entity that knows for sure if that segment has been updated. Therefore, the client nodes must decide, in function of the segment validity period, whether to use the segment that they have in memory or consult the CA if this CRL segment has been updated.

5. PERFORMANCE ANALYSIS

In our distributed system, CAs deliver new CRL segments to any node that sends a request to them. But from that time, these nodes can also deliver those segments to other nodes. Therefore, the main benefit of segmenting and distributing the segments of an entire CRL across different nodes is the reduction of storage requirements for each node and a corresponding reduction in the request rates. This reduction takes place both in CAs and in server nodes.

To analyze the request rate of the CAs, we define the probability density function with which a node sends a CRL segment request to a CA. Taking into account that CAs issue a new, or updated, CRL segment at time $t = 0$, we define the probability of a node requesting a CRL segment, for the first time, at time t.

If a CRL segment is issued at time $t = 0$, the probability that a node will send the CRL segment request within the interval $[t, t+dt]$ depends on the probability that the node will perform a validation within this interval. A node will request a certain segment within the interval $[t, t + dt]$ if and only if it attempts to validate a certificate that requires the use of that segment and has not validated any certificate that required the use of this segment within the interval $[0, t]$.

Since the number of nodes that usually make up the KAD network is large, we can assume that the carrying out time of certificate validation is mutually independent and follows the random distribution, which fulfills the Poisson Law. The probability of attempting n certificate validations in t is

$$\left[\frac{(vt)^n}{n!}\right] e^{-vt}, \quad n = 0, 1, 2, 3 \ldots, \tag{2}$$

where e^{-vt} is the probability that one node will not perform a validation within the interval $[0, t]$ and v is the validation rate (times of node's certificate validation per unit of time). In addition, we assume that all certificate validations are equally likely to require access to any of the CRL segments. If we have f CRL segments, there is a probability of $1/f$ that a certain segment i will be needed to perform any certificate validation. Thus, the probability that this segment will not be needed for any of n certificate validations is

$$\left(1 - \frac{1}{f}\right)^n. \tag{3}$$

Combining Equations (2) and (3), the probability that any node will not request segment i within the interval $[0, t]$ is

$$\sum_{n=0}^{\infty} \left(1 - \frac{1}{f}\right)^n \left[\frac{(vt)^n}{n!}\right] e^{-vt} = e^{-vt/f}. \tag{4}$$

The probability that a node needs the segment i within the interval $[t, t+dt]$, assuming that the probability of occurrence of more than one validation attempt is 0 since the interval $[t, t+dt]$ is infinitesimally small, is

$$ve^{-v\,dt}\,dt = v\,dt. \tag{5}$$

And as the probability that any validation will require the use of segment i is $1/f$, the probability that this segment will be needed in the interval $[t, t + dt]$ is

$$\frac{v\,dt}{f}. \tag{6}$$

Using Equations (4) and (5) and multiplying by the number of nodes (N), the total expected number of requests for the segment i within the interval $[t, t + dt]$ is expressed as

$$N_f(t) = \frac{Nv\,e^{-vt/f}\,dt}{f}. \tag{7}$$

And the total request rate for a new, or updated, CRL segment is

$$R_f(t) = \frac{f N_f(t)}{dt} = Nv\,e^{-vt/f}. \tag{8}$$

Taking into account that the request rate for entire CRLs from a CA at time t is $R(t) = Nv\,e^{-vt}$, the CRL segment request rate drops off with the number of segments in which the CRL is divided, but not so with the peak request, as $R_f(0) = R(0) = Nv$.

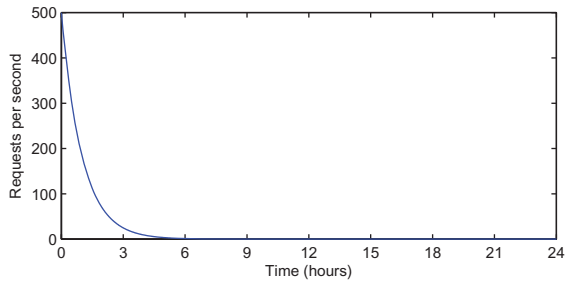

FIGURE 3. Traditional CRLs.

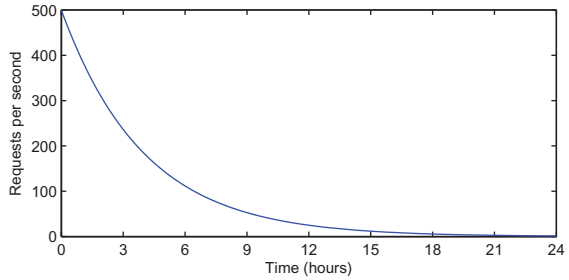

FIGURE 4. Segmented CRLs.

This difference can be seen by comparing Fig. 3 with Fig. 4. Figure 3 shows the request rate for an entire CRL and Fig. 4 the request rate for the CRL segments, both graphs over the course of 24 h. We have assumed that the entire CRL and the CRL segments were issued at time 0, and that no others were issued during the course of the 24 h. We have also assumed that the number of nodes within the KAD network N is 1 million, the validation rate (v) is 50 certificates per day and the number of CRL segments is $2^7 = 128$.

But in our distributed system, a CRL segment is also retrieved by several nodes. Therefore, we also determine the request rate of a certain node for a given segment. In addition, CRL segments are updated independently. To do so, we assume that all server nodes are selected with the same probability, as usually the nodes download the segments of the closest server node.

The probability that any node will not perform the request of the segment i to node j within the interval $[0, t]$ is

$$(N/P)\, e^{-vt(N/P)/f}, \tag{9}$$

where N is the number of nodes that perform a request, P is the number of potential server nodes of the CRL segment i and N/P is the average number of nodes that download the segment i of the same server node. The probability that one of these nodes will request the segment to node j within the interval $[t, t + dt]$ is

$$\frac{v\, e^{-v\, dt}\, dt}{f} = \frac{v\, dt}{f}. \tag{10}$$

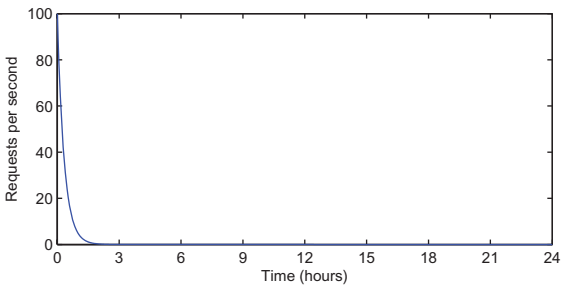

FIGURE 5. Distribution of segmented CRLs.

Combining Equations (9) and (10), we can determine the total expected number of requests for segment i to node j in the interval $[t, t + dt]$:

$$N'_f(t) = \frac{(N/P)v\, e^{-vt(N/P)/f}\, dt}{f}. \tag{11}$$

And the total request rate is

$$R'_f(t) = \frac{f\, N'_f(t)}{dt} = (N/P)v\, e^{-vt(N/P)/f}. \tag{12}$$

As can be seen in Fig. 5, the peak request has decreased, as $R'_f(0) = (N/P)v$, and the CRL request rate continues to decrease with the number of segments in which the CRL is divided. Thus, our system improves the distribution of CRLs compared with the traditional and fragmented CRLs methods.

In terms of performance of the client nodes, this system introduces a computational overhead when they must select a CRL segment, but it reduces the storage space that nodes must dedicate to the CRL. Nodes must calculate a modular hash function every time they need to validate a certificate; however, the time required to perform this operation with current computers is imperceptible to a user. In addition, they only need to store certain CRL segments and not the entire CRL. Regarding the CAs, the number of hash functions that they must calculate is very high, in particular one for each revoked certificate. However, these operations may be performed previously and stored in a database, so that the computational overhead would be negligible. Not so in the case of signatures, since the CAs must sign all issued and updated CRL segments. However, these signatures are not a problem because the segments are updated independently.

6. CONCLUSIONS

Today, structured p2p networks are not yet mature, in terms of security, to implement commercial applications such as paid video streaming applications. To address this problem, the adoption of PKI seems to fulfill the security requirements,

although there are some associated problems in these networks, such as the distribution of CRLs. The CAs can become a bottleneck, as the size of CRLs grows exponentially with the number of network users and the CRLs must be updated more frequently to maintain data freshness. For these reasons, we propose a new distributed revocation system for the KAD network, where the CAs split the entire CRLs into several CRL segments which are stored in a large number of nodes to improve accessibility and availability. In addition, these segments can be issued independently, which improves data freshness without incurring a great cost for the network.

The distribution and replication of the CRL segments within a p2p overlay decreases the peak request rate, avoiding bottleneck problems in the server nodes and improves the availability of the revocation data. While the segmentation of CRLs does not reduce the peak request rate, it reduces the size of the files that nodes must store. The nodes only need to download the CRL segment which contains the serial number of the certificate that they need to validate, if this has been revoked. Moreover, the server nodes can service requests at a faster rate.

Regarding the possible implementation of this distributed revocation system in other famous p2p networks, such as BitTorrent or P2PStream, in today's Internet, it is necessary to take into account a series of network features. All users' data must be stored by the owners, while pointers or references to them must be stored by the network. A chord network is a network example that does not adhere to this requirement, as nodes store the content of other owners. Moreover, it is important that the network replicates the pointers or references in a considerable amount of nodes. In this way, the searches are faster and the requests are more distributed. Finally, the search process and how resources are identified within the network should not be a problem; they just have to be adapted to the desired network.

FUNDING

This work was supported partially by the Spanish Research Council with Project SERVET TEC2011-26452, by Spanish Ministry of Science and Education with Project CONSOLIDER CSD2007-00004 (ARES) and by Generalitat de Catalunya with Grant 2009 SGR-1362 to consolidated research groups.

REFERENCES

[1] Aikebaier, A., Enokido, T. and Takizawa, M. (2011) Trustworthy group making algorithm in distributed systems. *Hum. Centric Comput. Inf. Sci. (HCIS)*, **1**, 1–15.

[2] Elmisery, A.M. and Botvich, D. (2011) Enhanced middleware for collaborative privacy in IPTV recommender services. *J. Conv. (JoC)*, **2**, 33–42.

[3] Teraoka, T. (2012) Organization and exploration of heterogeneous personal data collected in daily life. *Hum. Centric Comput. Inf. Sci. (HCIS)*, **2**, 1–15.

[4] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R. and Polk, W. (2008) Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard).

[5] Myers, M., Ankney, R., Malpani, A., Galperin, S. and Adams, C. (1999) X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 2560 (Proposed Standard).

[6] Wei, Q., Qin, T. and Fujita, S. (2011) A two-level caching protocol for hierarchical peer-to-peer file sharing systems. *J. Conv. (JoC)*, **2**, 11–16.

[7] Gnutella Home Page. http://rfc-gnutella.sourceforge.net (accessed August 5, 2012).

[8] Ying, G. and Jiang, Z. (2009) Research on CRL Distribution in P2P Systems. *Proc. 2nd IEEE Int. Conf. Computer Science and Information Technology. ICCSIT' 09*, Beijing, China, August 8–11, pp. 574–577. IEEE Computer Society.

[9] Stoica, I., Morris, R., Karger, D., Kaaskoek, M. and Balakrishman, H. (2001) Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. *Proc. ACM Conf. Special Interest Group on Data Communication. SIGCOMM' 01*, San Diego, CA, USA, August 27–31, pp. 149–160. ACM.

[10] Huang, J., Wang, Z., Qiu, Z. and Chen, M. (2008) Theoretical Analysis of Issuing Mechanism in Distributive Digital Certificate Revocation List. *Proc. Int. Conf. Computer and Electrical Engineering. ICCEE' 08*, Phuket, Thailand, December 20–22, pp. 199–203. IEEE Computer Society.

[11] Morogan, M. and Muftic, S. (2003) Certificate Revocation System Based on Peer-to-Peer CRL Distribution. *Proc. 9th Int. Conf. Distributed Multimedia Systems. DMS' 03*, Miami, FL, USA, September 24–26.

[12] Avramidis, A., Kotzanikolaou, P., Douligeris, C. and Burmester, M. (2012) Chord-PKI: a distributed trust infrastructure based on P2P networks. *Comput. Netw.*, **56**, 378–398.

[13] Beth, T., Borcherding, M. and Klein, B. (1994) Valuation of Trust in Open Networks. *Proc. European Symp. Research in Computer Security. ESORICS' 94*, Brighton, UK, November 7–9, pp. 3–18. Springer.

[14] Chen, R. and Yeager, W. (2002) Poblano: A Distributed Trust Model for Peer-to-Peer Networks. JXTA Security Project White Paper. Sun Microsystem.

[15] Abhilash, G. and Yoon, J.P. (2004) Modeling Group Trust for Peer-to-Peer Access Control. *Proc. Int. Workshop on Database and Expert Systems Applications*, Zaragoza, Spain, August 30–September 4, pp. 971–978. IEEE Computer Society.

[16] Maymounkov, P. and Mazières, D. (2002) Kademlia: A Peer-to-peer Information System Based on the XOR Metric. *Proc. Int. Workshop on Peer-To-Peer Systems. IPTPS' 02*, Cambridge, MA, USA, March 7–8, pp. 53–65. Springer.

[17] eMule Home Page. http://www.emule-project.net (accessed August 5, 2012).

[18] aMule Home Page. http://www.amule.org (accessed August 5, 2012).