Fast access control decisions from delegation certificate databases

Tuomas Aura *

Helsinki University of Technology, Digital Systems laboratory
FIN-02015 HUT, Finland; Tuomas.Aura@hut.fi

Abstract In new key-oriented access control systems, access rights are delegated from key to key with chains of signed certificates. This paper describes an efficient graph-search technique for making authorization decisions from certificate databases. The design of the algorithm is based on conceptual analysis of typical delegation network structure and it works well with threshold certificates. Experiments with generated certificate data confirm that it is feasible to find paths of delegation in large certificate sets. The algorithm is an essential step towards efficient implementation of key-oriented access control.

1 Introduction

In new key-oriented, distributed access control systems, entities are represented by their cryptographic signature keys. Authority is delegated from key to key by issuing signed delegation certificates. The certificates form graph-like networks of delegation between the keys [1]. Access control decisions in the system are made by verifying that there exists a path of delegation all the way from the server to the client.

The most prominent proposals for distributed trust management are SPKI certificates [3] by Ellison et al., SDSI public key infrastructure [6]) by Rivest and Lampson, and PolicyMaker local security policy database [2] by Blaze et al. The algorithms presented in this paper are directly applicable to SPKI-type certificates.

If key-oriented access control is to gain popularity, certificate management must be arranged in an automated and efficient manner. Specialized server software is needed for certificate acquisition, updates, bookkeeping and verification. These servers will need algorithms for finding delegation paths from large sets of certificates.

In this paper, we present an efficient technique for making authorization decisions from large databases of certificates. The basic idea is to start searches for a delegation path from both the server and the client keys and to meet in the middle of the path. The performance of the algorithm is assessed theoretically and experimentally based on general properties that we expect a typical delegation network to have.

* This work has been funded by Helsinki Graduate School in Computer Science and Engineering (HeCSE) and supported by research grants from Academy of Finland.

We begin by introducing delegation certificates in Sec. 2 and the problem of making authorization decisions in Sec. 3. Sec. 4 discusses typical properties of delegation networks. In Sec. 5, we describe threshold certificates and their role in the decision problem. Simple path-search algorithms for authorization decisions are presented in Sec. 6 and an efficient two-way algorithm in Sec. 7. Sec. 8 discusses the efficiency of the algorithms and Sec. 9 gives some experimental results. Sec. 10 concludes the paper.

2 Delegation network

A *delegation certificate* is a signed message with which one entity delegates some access rights to another entity. In a key-oriented system, the entities are represented by their private-public key pairs. The key signing the certificate is called the *issuer* and the key receiving the delegated rights is the *subject* of the certificate. Special *threshold certificates* may have several subjects that must co-operate in order to use the authority given by the certificate (see Sec. 5).

The subject can redelegate the access rights. This way, certificates form a network of delegation relationships between the keys. A key can authorize another to an operation directly by issuing a certificate to it or indirectly by completing a longer chain of certificates conveying the rights between the keys. The delegation is, of course, meaningless unless the issuer itself has the right to the delegated operations. It may, nevertheless, be useful to delegate rights that one expects to obtain in the future.

A client is entitled to request an operation from a server if there exists a chain of certificates in which the first certificate is issued by the server itself or by a key in its access control list, the client is the subject of the last certificate, and all certificates in the chain authorize the requested operation. Often, the client will attach the chain certificates to its request.

We think of the access rights as sets of allowed operations. Thus, the set of rights delegated by the chain of certificates is an intersection of the rights delegated by the individual certificates. On the other hand, if there are several independent or partially intersecting chains to the same subject from the same original issuer, the set of rights delegated by these chains is the union of the rights delegated by the individual chains.

We can visualize a database of certificates as a directed graph where keys and certificates are nodes and the arcs point in the direction of the flow of authority i.e. from the issuers to the certificates and from the certificates to the subjects. Fig. 1 shows a delegation network as a directed graph where the certificates have been annotated with the delegated operations and an equivalent simplified drawing. In the figure, there is, for example, a delegation path from key k_3 to key k_6 passing right for operations r and w.

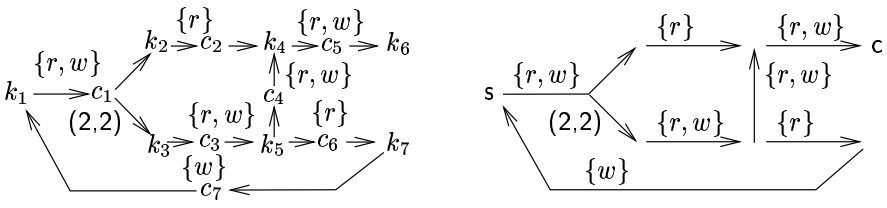

Figure1. A delegation network as a graph and a simplified drawing

3 Making authorization decisions

When the number of certificates possessed by an entity becomes large, it is not a trivial matter to decide who is authorized to which operation by whom. This task should be given to specialized certificate managers [2,4] that provide the authorization decisions (for servers) or certificate chains (for client) on request. The servers can also keep track of expiring or revoked certificates and they may automatically acquire new ones. This kind of arrangement, however, means that the certificate management becomes critical for the performance of the system. Consequently, fast algorithms will be needed for authorization decisions.

The decision algorithm should return a result after a finite execution time. In a finite database of certificates, the authorization problem is decidable but, unfortunately, in an enumerable infinite database, the authorization relation is only recursively enumerable. In the latter case, we must in some other way limit the search space to finite.

As surprising it may first sound, the question of finiteness is of great practical importance. If certificates are retrieved on demand from other servers during the decision procedure, the number of certificates to be examined may well have no upper bound. In such cases, however, the cost of certificate acquisition dominates the total expenses. Thus, we consider only algorithms for finite networks and occasionally mention the possibility of terminating unsuccessful searches after some maximum effort. Efficient strategies are also needed for retrieving certificates from the network but that is outside the scope of this paper.

4 Models of typical delegation network

In order to device efficient algorithms for authorization decisions, we need to have an understanding of the typical structure of the graphs formed by certificates. We anticipate that most delegation networks will have a statistically distinguishable layer structure with hourglass shape and different branching factor in forward and backward direction. Nevertheless, we will emphasize more the generality of the algorithms than their performance in particular situations.

The layered structure means that the access rights are distributed by a network of intermediate keys. There can be a single broker for delivering access rights from servers to clients as in Fig. 4(a), or more layers of keys as in 2 (a).

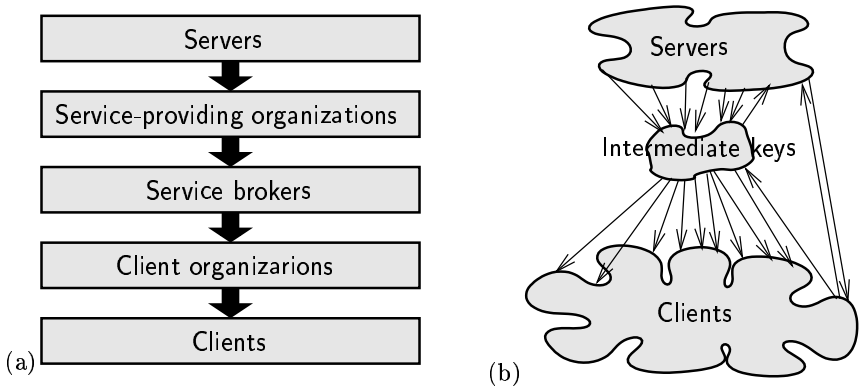

Figure2. Layered structure and hourglass shape of delegation networks

Usually access to the physical servers of a single service provider will be controlled centrally by the keys of the service-providing organization. Likewise, the client organizations will centralize the acquisition of access rights into a few certificate databases. In the middle, there may be service brokers as in most models of distributed computing (for example, [5]). With time, the roles of the intermediate keys are likely to become more fixed and the layered structure more apparent.

One consequence of the layering of keys is that the access rights will be passed from a large number of server keys through relatively few intermediate keys to a large number of client keys. That is, the network will be wide at both ends and narrow in the middle. This hour-glass shape is emphasized in Fig. 2 (b).

Naturally, the common structure will only hold for a majority of the certificates. There may be occasional short links, certificates issued by clients to servers, and relationships amongst servers or clients themselves. We will optimize the efficiency of our algorithms with the hourglass structure in mind but still accommodate certificates between arbitrary keys.

In most cases, there will be more clients than servers. Thus, on the average, the number of clients connected to one server must be larger than the number of servers connected to one client. At the local scale, we can measure the branching of the network in forward and backward directions: how many certificates does an average key issue and in how many certificates is an average key a subject. Although the branching factors will change as a function of the service and of the distance from the issuer, we will use a single branching factor in each direction for theoretical comparison of algorithms.

5 Threshold certificates

A normal certificate gives the access rights unconditionally to a single subject. Sometimes we want to grant the access rights to a group of keys that must

decide on their use together. This is accomplished with threshold certificates. A (k, n)-threshold certificate has a set of n subjects and a threshold value k ($\leq n$). The certificate is considered valid only if at least k of its n subjects co-operate in using the access rights. In practice, the k subjects will (directly or indirectly) delegate the authority to a single key that can then make requests to the server. In algorithms, we treat all certificates equally and view the normal (i.e. non-threshold) ones as having threshold value $(1, 1)$.

Threshold certificates complicate substantially the process of making authorization decisions because instead of a simple path of certificates, paths through the threshold number of subjects must be found in the delegation network.

For example, in the delegation network of Fig. 1, key k_1 authorizes k_6 to operation r because both k_2 and k_3 pass the right forward, but not to operation w because k_2 does not co-operate in redelegating it.

6 Simple path-search algorithms

In the literature, no algorithms for authorization decisions from a database of certificates have been described. The SPKI document [3] suggests certificate reduction as a decision making procedure. That is, rules are given for how two certificates forming a chain can be reduced into a single new certificate. A server should grant access to a client if there is a path of certificates that reduces into a single certificate where the server or a key in the server's access control list directly authorizes the client. The problem with reduction as an implementation technique is that someone must decide which certificate chain to reduce. Thus, an efficient way of finding complete chains in a delegation network is needed in any case. Some discussion on the implementation techniques for policy managers but no complete algorithms can be found in [4].

We can use simple path-finding algorithms in the graph of keys and certificates to find the paths of delegation. The most straight-forward choice would be a depth-first search (DSF) from the server for the client.

For delegation networks with no threshold certificates, the DFS algorithm runs relatively fast. The complexity is linear with the size of the network. With threshold certificates, the search becomes hopelessly inefficient. This is because the search must process some parts of the network two or more times. Fig. 3 (b) shows a simple example. There is a path from the key k to the client. The first visit to key k by DFS, however, does not discover this path because it would form a loop with the path from the client to k. Failure on the other subject of the threshold certificate causes the first found path to c to be rejected. Consequently, the paths from key k must be reconsidered on the second visit. (This is a good test case for algorithmic improvements.) When the number of threshold certificates increases, the reprocessing of nodes takes an exponential time. Typically, existing certificate paths are found in reasonable time but negative answers can take millions of steps.

In addition to retraversing paths, the forward search has another inefficiency. In Fig. 4 part (a), there is an hourglass-shaped delegation network with larger

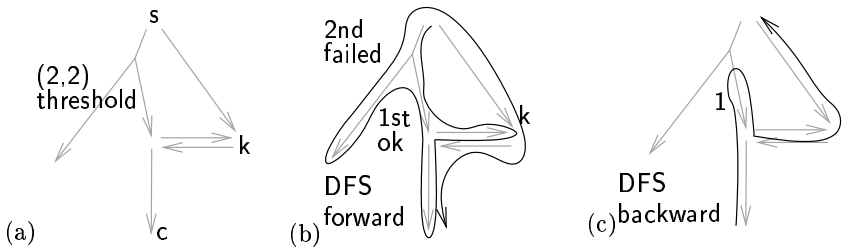

Figure3. Depth-first forward search can visit the same node several times.

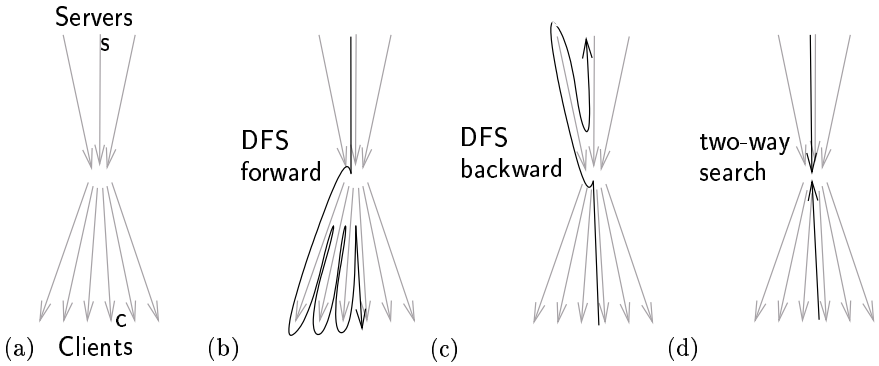

Figure4. Searching backward or in two directions is faster.

average branching factor in the forward direction. Part (b) shows how a forward depth-first search from the server finds the client. In part (c), the search is initiated backward from the client to find the server. The searches are functionally equivalent, but since the network branches less in the backward direction, the backward search is faster.

Backward search can be done in depth-first or breadth-first order. Since both searches mark found keys, their space requirement is linear with the number of visited keys. The advantage of BFS is that the paths are explored by increasing depth. This may be useful if the certificate database is too large for an exhaustive search of paths. In that case, the maximum depth of the search can be determined dynamically during the search and an unsuccessful search can be terminated at the same depth in all directions.

For threshold certificates, BFS counts the number of paths found from the client to the subjects. The search continues backward from the issuer of the threshold certificate only when the threshold value is reached. Fig. 3 (c) illustrates how the backward search processes every key at most once.

To summarize, the backward search in a delegation network with many threshold certificates is much more efficient than the forward search. It takes a linear time with respect to network size. Because of the different branching

factors, it is also more efficient in a delegation network where all certificates have a single subject.

7 Two-way search

Although the previous section showed that the graph search is more efficient in the backward direction, we will not completely discard the forward search. Instead, we will optimize the process by combining searches in both directions. The idea is to start the search from both the client and the server and to meet in the middle of the path. This is illustrated in Fig. 4 (d). When the delegation network has only few middle nodes and the average branching factors are large, searching from two directions can radically improve the performance.

Listing 1 gives pseudocode for the forward search part. It marks keys until it terminates at a specified maximum depth or at a threshold certificate. After the forward search, we do a backward search from the client to find a marked key. In Listing 2, there is pseudocode for the backward search algorithm.

The searches in the two directions can also be done interleaved or in parallel. This would make it possible to continue the search dynamically from the side where the branching is smaller. This may be a good solution for networks with unknown structure. When the layer structure is known accurately (like in our experiments), there is usually a clearly distinguishable optimum meeting depth.

```
1  function DFSforward (key,client,operation,depth)
2      if (key marked as found by DFS) return FALSE;
3      mark key as found by DFS;
4      if (key = client) return TRUE;
5      if (depth = 0) return FALSE;
6      for c in certificates issued by key
7          if (number of subjects of c = 1 AND
8              c authorizes operation AND
9              DFSforward(subject of c,client,operation,depth-1))
10             return TRUE;
11     return FALSE;
```

Listing 1: DFS forward to mark keys to a given depth from the server

8 Efficiency estimates

We justify the two-way search by performance comparison in two simplified models of typical delegation network (based on Sec. 4). We first consider an infinite network with homogeneous nodes and then refine the model by dividing the keys to layers. We assume that the delegation network is infinite, that already visited keys are never met again, that a path is always found or the search is

```
12 function BFSbackward (client,operation)
13     nextKeys = {client};
14     mark client as having path to client;
15     while (nextKeys != ∅)
16         currentKeys = nextKeys;
17         nextKeys = ∅;
18         for key in currentKeys
19             for c in certificates given to key
20                 if (c authorizes operation AND
21                     issuer of c NOT marked as having path to client)
22                     countPaths = 0;
23                     for subj in subjects of c
24                         if (subj marked as having path to client)
25                             countPaths = countPaths + 1;
26                     if (countPaths ≥ threshold value of c)
27                         if (issuer of c marked as found by DSF)
28                             return TRUE;
29                         if (exists a certificate given to issuer of c)
30                             nextKeys = nextKeys ∪ {issuer of c};
31                         mark issuer of c as having path to client;
32     return FALSE;
```

Listing 2: BFS backward from client to a marked key

terminated at a specified depth, and that there are no threshold certificates. (The net effect of threshold certificates would be to make forward search more difficult.)

We first consider a network where all keys are alike. A key issues, on the average, β_f certificates and is the subject of β_b certificates that transfer rights to an operation o.

The average cost of finding a path delegating the right to an operation o between two nodes in the graph grows exponentially with the length of the path. The number of keys visited on an average backward search to depth d is

$$1 + \beta_b + \beta_b^2 + \ldots + \beta_b^d \;=\; (\beta_b^{d+1} - 1)/(\beta_b - 1) \;=\; O(\beta_b^d). \qquad (1)$$

By searching from both ends and meeting in the middle, we can in the best case reduce the problem to two parts with path length $d/2$. If the branching factors in both direction are equal, the complexity decreases to approximately the square root of the original, $O(\beta_b^{d/2})$.

When the branching factors in the two directions differ, the efficiency improvement is not quite as great. The reason is that a one-way search is always done in the direction of the smaller branching factor while a two-way search must also go in the less beneficial direction. The meeting point should be set nearer the end from which the branching is greater, not half-way between. If the expected distance d and the branching factors β_f forward and β_b backward are large enough, the optimal meeting distance from the server is $d \log \beta_b / (\log \beta_f + \log \beta_b)$. The searches either have to proceed in parallel to keep the depth ratio of the

forward search to the backward search at the optimal value $\log \beta_b : \log \beta_f$, or we can insert a measured average distance d to the above formula and stop the forward search at the precalculated depth.

# of certificates per *issuer*	from layer 1	2	3	4	5
1	0	0	0	0	0
2	1	2	0	0	0
to layer 3	1	2	2	0	0
4	2	2	5	2	0
5	10	5	10	20	0

layer	relative amounts of keys
n_1	10
n_2	5
n_3	2
n_4	20
n_5	2000

# of certificates per *subject*	from layer 1	2	3	4	5
1	0	2	5	1	0.05
2	0	2	5	0.5	0.0125
to layer 3	0	0	2	0.5	0.01
4	0	0	0	2	0.2
5	0	0	0	0	0

Table 1: A forward and backward branching matrices

A more realistic model should divide the keys into several layers. Branching in this kind of network can be described with two *branching matrices*. The forward branching matrix B_f tells how many certificates an average key in each layer issues to other layers. The backward branching matrix B_b tells how many certificates an average key in each layer receives from other layers. For example, the forward branching matrix in Table 1 says that a layer-3 key issues on the average 5 certificates to keys in layer 4 for authorizing the operation in question.

The forward and backward branching matrices can be calculated from each other if the relative distribution of keys to the layers is known. Let n_i and n_j be the relative amounts of keys in layers i and j. The elements of the forward and backward matrices are related by the formula

$$n_i \cdot B_f^{(j,i)} \;=\; n_j \cdot B_b^{(i,j)}. \tag{2}$$

The backward matrix in Table 1 calculated from the forward matrix and the key distribution.

The branching matrices give the average number of certificates issued and received per key. The distribution of the certificates between the keys inside the layers may vary. This does not affect the average number of keys visited in the search but it does affect the distribution of search sizes.

Analogously to Formula 1, total number of keys visited by an average backward search to depth d is

$$\begin{aligned} &(I + B_f + B_f^2 + B_f^3 + \ldots + B_f^d) \cdot [0\ 0 \ldots 0\ 0\ 1] \\ &= [1\ 1\ \ldots 1\ 1\ 1](B_b - I)^{-1}\ (B_b^{d+1} - I)\ [0\ 0 \ldots 0\ 0\ 1]^T. \end{aligned} \tag{3}$$

Again, we make the assumption that already visited keys are never found again. The constant vectors must have as many elements as there are layers of keys. 1 in the last place of the other vector indicates that the backward searches are started from a key in the client layer.

The value of the formula converges when the spectral radius (maximum absolute eigenvalue) of the matrix B_b is smaller than 1. This is so if the lower half

and the diagonal of the matrix are all zeros, meaning that the rights are always passed towards the client and never backwards or to the same layer. In that case, the search will terminate even in an infinite delegation network.

In a two-way search, we again divide the task to two smaller ones with depths around $d/2$ which means significant performance improvement. The optimal meeting depth is somewhat more complicated to estimate. One way is to calculate the total number of keys visited by forward and backward searches at increasing depths and to select the depths in both directions so that their sum is sufficient but the sum of the visited keys still affordable.

In the example of Table 1, the numbers of keys visited by forward searches to depths 0,1,2,3, ... are

$$1, 15, 87, 419, 1679, 5831, 18279, 53223, 146727, 388007, 993191, \ldots$$

and by backward searches

$$1, 1, 2, 6, 19, 60, 182, 517, 1404, 3679, 9354, \ldots$$

If we can afford to visit up to 1000 keys, we should go to depth 4 in the forward direction and to depth 8 in the backward direction. ($419 + 517 \approx 1000$) That way, we can discover all paths up to length 12. Paths of length 4 can be found in only 19 steps searching backwards.

Although the two-way search performs better in our ideal models, the results are not accurate for exhaustive searches where, in the end of an unsuccessful search, a significant portion of the keys may have already been marked as seen. Therefore, we present experimental evidence in the next section.

9 Experimental results

Experiments conducted with generated certificate data confirm that the two-way search is more efficient than one-way search algorithms. The backward searches also perform reasonably well. Forward search is very slow when threshold certificates and backward links are present.

Since no real-world certificate databases are available for the time being, we generated random delegation networks with the assumed layered hourglass structure. The number of keys on each layer and of certificates between each two layers were chosen according to our idea of the typical system. The network was then constructed by creating the certificates between random keys.

The data presented here was collected from a network with 4 layers of keys. Table 2 shows the number of keys in each layer and the total number of certificates between each two layers. It should be noted that in this network, there are only few backward arcs towards the server. (This is determined by the upper half of the matrix giving the certificate counts.)

The experiments with different *one-way* algorithms showed that the backward DFS and BFS perform best (see Table 3). Any performance differences between these two algorithms were insignificant and certainly much smaller than differences caused by implementation details. As expected, DFS performed badly.

Layer	# of keys
1	100
2	10
3	100
4	5000

Total # of certificates		from layer 1	2	3	4
	1	5	2	2	2
to layer	2	200	2	2	2
	3	10	200	5	2
	4	100	10	20000	500

# of subjects	% of certificates
1	80
2	15
3	3
4	2

Table 2: Parameters for the generated delegation network

Decision	Search algorithm DFS forward	DFS backward	BFS backward
all	3273	56	54
positive	3581	53	51
negative	2347	64	64

Table 3: Average number of algorithmic steps for a key pair in different algorithms

The results of the comparisons between algorithms were relatively stable with small changes in the parameter values except for the forward search. In the delegation network of Table 2, the forward search took about 50 times more time than the pure backward searches. The efficiency of the forward search is greatly dependent on the frequency of threshold certificates, on the completeness of the graph and on the number of backward arcs from layers near the client to layers near the server. These arcs create more paths in the graph, and the depth-first forward search may traverse a lot of them. The positive answers are usually returned quite fast while negative results may require exponentially more work. In some networks, the forward searches become painfully slow taking occasionally millions of steps to complete queries with a negative result.

Decision	Depth of forward search 0	1	2	3	4
all	56	42	67	1517	1606
positive	51	32	58	1714	1804
negative	70	71	92	970	1053

Decision	Depth of forward search 0	1	2	3	4
all	58	36	73	1900	1895
positive	50	21	60	2065	2048
negative	81	82	116	1370	1406

Table 4: Cost of two-way search with (left) and without threshold certificates

The two-way search was tested by first starting depth-first forward from the server to a specified maximum depth or until threshold certificates were met, and then looking for the marked nodes with breadth-first search backward from the client (see Sec. 7). Table 4 (left side) shows how the cost of the computation varied in the two-way search as a function of the depth of the forward search.

In the delegation network of four layers, one step of forward search gave the best results. The average savings amount to 25 %. With other network parameters, the best results were also given by forward search to the depth of 1 to 3 certificates. When the network had five to six layers of keys, a forward

search of depth two was fastest. The savings in computation time varied between 10 and 50 % being better when the answers were mostly positive.

Table 4 also shows the same measurements for a network without any threshold certificates. Here the two-way search saves about 60 % of the cost for queries where a valid path is found. The performance improvement is bigger because the forward search part in the two-way search stops at threshold certificates.

Altogether, the results suggest that a two-way search with a fixed meeting depth of 2 from the server works well in most cases. Parallel search from the two directions with a dynamically determined meeting depth might perform slightly better but it is more difficult to implement.

10 Conclusion

We described an efficient two-way search technique for making authorization decisions from certificate databases. The algorithm is based on a combination of depth-first and breadth-first searches that have been enhanced to handle threshold certificates. Conceptual analysis of typical delegation network structure and measurements on generated certificate data were done to assess the efficiency of the algorithm. The main observation from the experiments was that it is feasible to make authorization decisions from large delegation networks comprising thousands of keys and certificates. The algorithm for authorization decisions is an essential step towards efficient certificate management for key-oriented access control systems.

References

1. Tuomas Aura. On the structure of delegation networks. In *Proc. 11th IEEE Computer Security Foundations Workshop*, Rockport, MA, June 1998. IEEE Computer Society Press.
2. Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proc. 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, CA, May 1996. IEEE Computer Society Press.
3. Carl M. Ellison, Bill Franz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylönen. SPKI certificate theory, Simple public key certificate, SPKI examples. Internet draft, IETF SPKI Working Group, November 1997.
4. Ilari Lehti and Pekka Nikander. Certifying trust. In *Proc. 1998 International Workshop on Practice and Theory in Public Key Cryptography PKC'98*, Yokohama, Japan, February 1998.
5. Thomas J. Mowbray and William A. Ruh. *Inside Corba : Distributed Object Standards and Applications*. Addison-Wesley, September 1997.
6. Ronald L. Rivest and Butler Lampson. SDSI — A simple distributed security infrastucture. Technical report, April 1996.