Comparison of graph-search algorithms for authorization verification in delegation networks

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

Abstract

We describe and compare several algorithms for authorization decisions from a database of certificates. The algorithms are based on well-known graph-search techniques that we enhance to handle joint-delegation certificates. Experiments on generated certificate data were done to compare the efficiency of the algorithms.

1 Introduction

Emerging key-based access control mechanisms are moving the access control decisions from centralized trusted servers to the local ones. They also shift the focus from identity of entities to authorization, the right to perform operations. The access right can be delegated anonymously from key to key with a chain of certificates. Three key-based access control systems have received wide attention: SPKI certificates [2], SDSI public key infrastructure [3], and PolicyMaker local security policy database [1]. The authorization decision in all of these systems is to some extent dependent on the chaining of signed certificates that grant authority to perform operations. Thus, they must have some mechanism for determining if a set of certificates contains a chain that authorizes an operation.

In this paper, we present and compare algorithms for authorization decisions from sets of certificates. We take a simplified view of certificates: A certificate is issued by a key. With the certificate, the issuer authorizes another key, the subject of the certificate, or a group of subject keys together, to perform an operation and to delegate this right to other keys. We view the certificates as arcs of a generalized directed graph, delegation network. Our algorithms are based on ideas from graph search.

We believe efficient algorithms of this kind to be necessary when the distributed access control systems are implemented. The tasks of making authorization decisions and maintaining the certificates is likely to be given to a specialized servers that must have predictable performance. Also, if common-place applications such as databases and file and document servers start using the new access control systems, the volume of the certificate data may become so high that the techniques for its processing should be carefully selected. Experimental results with certificate data generated according to our understanding of a typical delegation network structure show that the authorization decisions can be made efficiently from large and variable sets of certificates as long as some care is taken in designing the algorithms for the purpose.

We begin by introducing our abstract view of the delegation networks and the authorization problem in Sec. 2. Sec. 3 describes the algorithms. In Sec. 4 we give results of experiments and comparisons between the algorithms. Sec. 5 concludes the paper.

2 Delegation network

We first define the terminology and authorization problem and then proceed to describe what the typical delegation networks look like.

2.1 Definitions

In our discussion, a *certificate* is a four-tuple $< issuer, subjects, k, authorization >$. $issuer$ is the key signing of the certificate and $subjects$ is a set of n keys to whom the certificate has been given. k is the *threshold* value determining how many subjects must co-operate to use and further delegate the right specified by $authorization$.

A *delegation network* is a set of certificates. The issuers and subjects in the certificates are called *keys*. A delegation network can be stored as a (generalized) directed graph structure where the certificates serve as directed arcs and keys as nodes. The joint-delegation certificates can be viewed as generalized arcs. If all certificates have only a single subject, the delegation network becomes a standard directed graph.

In our simplified model, delegation is always transitive. (In actual systems, the transitivity can be limited. In SPKI, for example, further delegation can be forbidden in the certificate. The algorithms in this paper are easily adaptable to such limitations.)

The authorization problem can be formulated as the existence of a computation [2]. Instead, we will try to help the reader visualize the problem with the help of trees. This is important because in order to understand the algorithms in this paper, the delegation network should be visualized as a graph. We also assume that the certificate database is set up as a graph so that the keys and their certificates are easily accessible from one another.

The tree-based definition goes as follows: A delegation network authorizes a client key c to perform an operation for a server key s if a finite tree can be formed such that

1. All nodes of the tree are marked with a key. The key on the root node of the tree is s. The key on every leaf node of the tree is c. The same key can be used for multiple nodes.

2. All nodes that have children (i.e. non-leafs) are marked with a certificate. The issuer of the certificate must be the key on that node. The subjects of the certificate must correspond to the keys on the children. If the threshold value of the certificate is lower than the number of subjects, only the threshold number of children are required. The same certificate can be used for multiple nodes. For a certificate with only a single subject, this means that there is a single child node and the key on the child node is the same as the subject. (Actually, we are not as much marking the nodes with certificates but the connection between the parent node and the group of children.)

Thus, the question "Does a delegation network authorize key c to perform operation s from server s?" can be stated as: "Does a tree exist that conforms to the above requirements?" With the graph-search algorithms, we are trying to find this kind of tree in the generalized

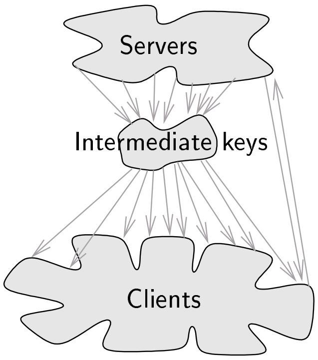

Figure 1: Typical delegation network structure

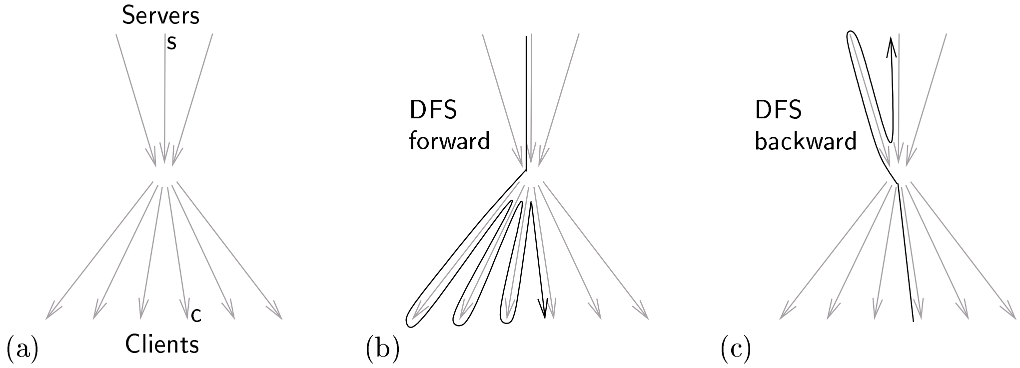

Figure 2: When forward branching is greater, backward search is faster

directed graph formed by the keys and certificates. When there are no joint-delegation certificates, the tree is reduced to a path, and standard path-finding algorithms for directed graphs can be used.

2.2 Typical delegation network structure

The delegation networks in practice, however, will not be arbitrary graphs but they will have certain structure. Although the system architectures themselves do not constrain the associations between the keys, common practices will arise from the way popular applications choose to chain their certificates.

We anticipate that most delegation networks will have an hourglass shape (Fig. 1). On the top of the hourglass there are the servers and on bottom the the clients. Direct certificates between the servers and clients are scarce. Instead, the access rights are distributed to the clients by a network of intermediate keys. These can be trusted certificate databases near the clients, reference monitors near the servers, and service brokers between them. In the extreme case, there could be a single broker delivering access rights from servers to clients, as in Fig. 2(a).

Application programs, user platforms and specialized servers themselves are unlikely to incorporate wide capabilities for maintaining valid certificates. Therefore, trusted servers are needed for certificate acquisition, updates, bookkeeping and verification. These servers will need algorithms for authorization decision from large sets of certificates, and they themselves form an additional key layer in the network.

Naturally, the common structure will only hold for majority of the certificates. There may be occasional short links and even certificates from clients to servers. Also, the servers or clients can create a wealth of mutual relationships amongst themselves. Thus, the system must be able to accommodate arbitrary certificates between arbitrary keys. Nevertheless, we will optimize the efficiency of our algorithms to with the hourglass structure in mind.

3 Algorithms for access control decision

In the literature, no actual algorithms for authorization decisions have been described. The SPKI document [2] explicitly states that its authors believe that the authorization questions can be answered but no implementation exists for the time being. In this section, we will shortly refer to the semantical definition of the SPKI certificates and then describe several algorithms for the authorization decisions.

Two things are worth noting about our algorithms. Firstly, they are based on simple path-finding algorithms for directed graphs. We have not considered any pre-computation techniques. Storing the precomputed results or some partial information in the memory can lead to constant-time algorithms but the memory space required is $O(n^2)$ with respect to the size of the certificate database. This does not seem feasible for the implementations that we have in mind, although some kind of caching might improve the efficiency of our algorithms. Secondly, signature verification is not part of the algorithm. All signatures are verified at the time when the certificates are entered into the database.

3.1 Five-tuple reduction

The SPKI document defines the semantics of the certificates with a five-tuple reduction. (The five-tuples are certificates almost like our four-tuples.) That is, rules are given for how two certificates (or more in the case of joint delegation) reduce into one. A server should grant access to a client if there is a path of certificates that recursively reduces to one certificate where the server itself authorizes the client.

The five-tuple reduction can be used as an implementation technique. When the client asks intermediate keys to sign the reduced certificates, no signature needs to be verified more than once. Still, the client, or an entity trusted by the client, must maintain the certificates and decide when it needs to start reducing a path. Therefore, techniques are needed for efficient path finding and decision making from a set of certificates even when the five-tuple reduction is actually implemented.

3.2 Depth-first search forward

The most straight-forward way to verify authorization from a certificate graph is depth-first search in the certificate graph tracing the flow of access rights from the server to the client. The recursive search procedure has, in fact, been proposed as an alternative semantic definition of authorization for the SPKI certificates [4].

Pseudo-code for a recursive depth-first search algorithm is listed in Listing 1. The algorithm should contain no surprises to the reader. For each certificate, it counts the valid

```
function dfsForward (server,client,operation)
    return dfsForwardRecursive(server,client,operation);

function dfsForwardRecursive (key,client,operation)
    mark key as in search path;
    if (key = client) return TRUE;
    for c in certificates signed with key
        if (c authorizes operation)
            countPaths = 0;
            for (subj in subjects of c)
                if (countPaths < number of paths required by c
                    AND (subj marked as having path to client
                         OR (subj NOT marked as having path to client
                             AND subj NOT marked as in search path
                             AND dfsForwardRecursive(subj,client,operation))))
                    countPaths = countPaths + 1;
            if (countPaths ≥ number of paths required by c)
                mark key as having path to client;
                return TRUE;
    unmark key as in search path;
    return FALSE;
```

Listing 1: Depth-first search forward from server to client

paths leading from subjects of the certificate to the client. If the count reaches the threshold required by the certificate (the threshold is 1 for non-joint delegation), there is a valid authorization path from the issuer of the certificate to the client.

Unfortunately, the number of paths in a graph grows exponentially with the graph size. Fig. 3(b) shows an example of how forward search must process some nodes again even though they have been visited before. (This is a good test case for algorithmic improvements.) If all certificates had only one subject, a linear algorithm could be used instead.

Our implementation that was used for the experiments reported in Sec. 4, does several further optimizations to avoid retraversing paths. Although these significantly reduce the number of keys processed, the complexity of the algorithm remains exponential. Typically, existing certificate paths are found fast but negative answers can take even millions of steps.

3.3 Depth-first and breadth-first search backward

In Fig. 2, there is a simple hourglass shaped delegation network. Part (b) shows how a forward depth-first search from the server finds the client. In part (c), the same kind of search is initiated backward from the client to find the server. The searches are functionally equivalent (even the same algorithm can be used when all certificates have only one subject), but since the network branches less in the backward direction, the backward search is faster. That is, in a typical delegation network, backward search will perform better than forward search.

There are two possible ways for handling joint-delegation certificates in backward search.

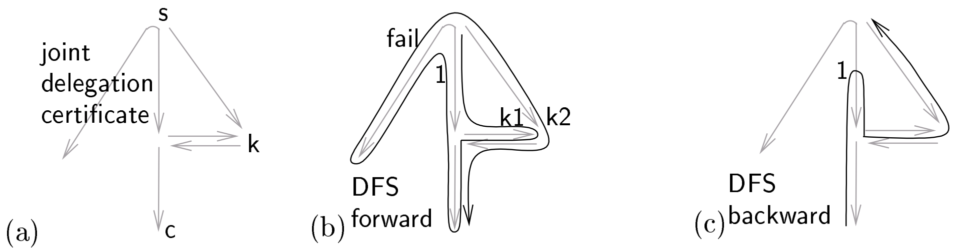

Figure 3: Depth-first forward search can visit the same node several times.

```
function dfsBackward (server,client,operation)
    return dfsBackwardRecursive(client,server,client,operation);

function dfsBackwardRecursive (key,server,client,operation)
    mark key as having path to client;
    if (key = server) return TRUE;
    for c in certificates given to key
        if (c authorizes operation
            AND issuer of c NOT marked as having path to client)
            countPaths = 0;
            for (subj in subjects of c)
                if (subj marked as having path to client)
                    countPaths = countPaths + 1;
            if (countPaths ≥ number of paths required by c
                AND dfsBackwardRecursive(issuer of c,server,client,
                                                    operation)
                return TRUE;
    return FALSE;
```

Listing 2: Depth-first search backward from client to server

One can span forward searches from the subjects in order to determine immediately if enough paths from the subjects to the client exist. This approach suffers from the poor performance of the forward search. Instead, we have chosen to count the number of paths leading from the client to the subjects, and to continue the backward search from the issuer of the joint-delegation certificate when the threshold value is reached. This appears to be simple and effective. If the same subject never appears twice in the same certificate, the counting can be optimized by keeping counters with the certificates. In the pseudocode of Listing 2, we have chosen the most general approach and recount the subjects on every visit. Fig. 3 illustrates how the backward search processes every key at most once.

Backward search can also be done in breadth-first order. The breadth-first algorithm processes keys by increasing distance from the client. Like in the depth-first algorithm, the issuers of joint-delegation certificates are discarded until enough of the subjects of the certificate have been processed. In theory the breadth-first search can require more memory than the depth-first search. In our experiments, however, the memory consumption was so small that we found it difficult to give any estimates.

```
function bfsBackward (server,client,operation)
    nextKeys = {client};
    mark client as having path to client;
    while (nextKeys = ∅)
    currentKeys = nextKeys;
        nextKeys = ∅;
        for key in currentKeys
            for c in certificates given to key
                if (c authorizes operation
                    AND issuer of c NOT marked as having path to client)
                    countPaths = 0;
                    for subj in subjects of c
                        if (subj marked as having path to client)
                            countPaths = countPaths + 1;
                    if (countPaths ≥ number of paths required by c)
                        if (number of certificates given to issuer of c > 0)
                            nextKeys = nextKeys ∪ {issuer of c};
                        mark issuer of c as having path to client;
                        if (issuer of c = server) return TRUE;
    return FALSE;
```

Listing 3: Breadth-first search backward from client to server

3.4 Two-way search

The graph search can be optimized by starting from both ends and meeting in the middle of the path. This is illustrated in Fig. 4.

The average cost c of finding the path between two nodes in a graph grows exponentially with the length of the path d. By searching from both ends and meeting in the middle, we can reduce the problem to two parts with path length $d/2$. This way, the complexity decreases to approximately the square root of the original. In normal graphs, we can search both ways and mark visited nodes along the way. When one search finds a node visited by the other, we know that a path exists. In order to find the complete path, the search that had first visited the node and already left it must retraverse the graph to find that node again, unless memory can be used to remember the paths to all visited nodes. When we do not have excess memory at disposal, the two-way search thus reduces the cost of deciding the existence of a path between two nodes to $2\sqrt{c}$ and the cost of finding the path to $3\sqrt{c}$. (This is, of course, not complete mathematical treatment but it should give an idea of the magnitude of the expected benefits.)

When the branching factors of the graph in the two directions are different, as in our case, the efficiency is not improved quite as much but still significantly. The reason is that one-way search is always done in the direction of smaller branching factor while two-way search must also go in the less beneficial direction. If the branching factors can be estimated, the meeting point should be set nearer the end from which the branching is greater, not half-way between. If the distance d and the branching factors β_1 and β_2 are large enough, the optimal meeting point is distance

$$d \log \beta_2 / (\log \beta_1 + \log \beta_2)$$

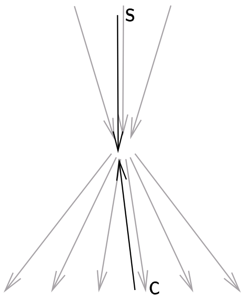

Figure 4: Two searches meet in the middle

away from the end from which the branching is β_1.

We implemented the two-way search by first doing depth-first search forward from the server, marking the visited keys on the way, and then trying to find a marked key with breadth-first (or depth-first) search from the client. The forward search algorithm is a simplified version of Listing 1 that ignores joint-delegation certificates and, thus, needs to visit every key only once. The forward search is terminated at a specified maximum depth. Experiment showed that in practice the above formula cannot be used to determine the optimal depth. Instead, a constant value of one or two should be used unless the delegation paths are especially long.

Unfortunately, the gains of two-way searching do not seem to be as big in practice as in the theoretical discussion above. The main reason is that the joint-delegation certificates make forward search from the server to the client almost infeasible. Therefore, it is not advisable to come more than one or two certificates away from the server to meet the backward search. Nevertheless, when large numbers of servers sign certificates for a few intermediate keys, the benefits of first marking keys one or two keys away from the server can be noticeable. If all certificates have only a single subject, the situation is quite different because also the forward search can be done more efficiently.

4 Experimental results

Experiments conducted with generated certificate data show that the gains from two-way search are not quite as big as expected. The backward search algorithms appear almost as efficient.

4.1 Generating certificate data

Since no real-world certificate databases are available for the time being, we generated random delegation networks with the assumed hourglass structure. This was done by dividing the keys to several levels, the top level representing servers and the bottom level clients. The number of keys on each level and the amounts of certificates between each two levels were chosen according to our (admittedly vague) idea of the typical system. The network was then automatically constructed by assigning the certificates between random keys in the specified levels.

The data presented here was collected from a network with 4 layers of keys. Table 1 shows the number of keys on each level and the certificates between them. It should be noted

Level	# keys	# certificates		from level				Number of subjects	% of certificates
				1	2	3	4		
1	100		1	5	200	10	100	1	80
2	10	to level	2	2	2	200	10	2	15
3	100		3	2	2	5	20000	3	3
4	5000		4	2	2	2	500	4	2

Table 1: Parameters for the generated delegation network

	Search algorithm			
Decision	dfs forward	dfs backward+forward	dfs backward	bfs backward
all	3273	4327	56	54
positive	3581	4210	53	51
negative	2347	4676	64	64

Table 2: Average number of algorithmic steps in for a key pair in different algorithms

that in our sample network, there are only few backward arcs towards the server. (This is determined by the lower half of the matrix giving certificate counts.) We found the results of comparisons between algorithms to be relatively stable with small changes in the parameter values. The amount of backward arcs and the arcs inside the levels, however seemed to have great effect on the efficiency of the forward depth-first search. Although we have chosen the parameters to somewhat help that algorithm, the results will not be too favorable in any case.

4.2 Results of comparison

The experiments with different one-way algorithms showed that the breadth-first backward search and depth-first backward search perform best (see Table 2). Any performance differences between these two algorithms were insignificant and certainly much smaller than differences caused by implementation details. The depth-first forward search and the depth-first backward search that spans forward searches at joint-delegation certificates, performed badly.

In the delegation network of Table 1, the forward searches took about 50 times more time than the pure backward searches. The efficiency of the depth-first search is greatly dependent on the degree of completeness of the graph and on the number of backward arch from levels near the client to levels near the server. These arcs create more paths in the graph, and the depth-first 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 negative result.

In the comparisons, the lookahead test of the breadth-first backward search (Listing 3, line 56) was disabled. We observed that the lookahead can reduce the number of keys processed in the algorithm by about up to 70 %. The more pure clients, i.e. keys that only receive certificates, there are, the more significant the speed-up will be. Hence, the optimization is in many situations more significant than it first seems.

Two-way search was tested by the first starting depth-first forward from the server to a

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

Table 3: Average number of algorithmic steps for a key pair in two-way search

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: Average cost in two-way search with no joint delegation

constant depth, and then looking for the marked nodes with breadth-first search backward from the client. (The depth-first search ignored all joint-delegation certificates; see Sec. 3.4.) Table 3 shows how the cost of computation varied in the two-way search as a function of the depth of the forward search.

The one-step forward search gave the best results. This is probably because that one step away from the client saves a lot of work in going through the large number servers attacked to a single broker. The savings amount to only 25 %. In experiments with other delegation network parameters, the best results were also given by forward search to the depth of one, or sometimes two, certificates. The savings in computation time were between 10 and 50 %. Thus, the two-way search does not perform as well as one would expect. It is also important to note that setting the maximum depth for the forward search to a number higher than 2 is risky: the complexity usually jumps up at depth 3 or 4. In delegation networks with significantly more than five distinguishable levels of keys, the desirable depth of the forward search could be higher, but we find such networks are unlikely to exits.

Table 4 shows the same kind of measurements as Table 3, only for a network without any joint-delegation certificates. Here we can see that the two-way search saves about 60 % of the cost for queries where a valid path is found. The performance improvement is much bigger than in the general delegation network. This is natural because the forward search part in the two-way search cannot handle well joint-delegation certificates.

It is also interesting to compare the last column of Table 2 with the first column of Table 3. These figures should be approximately equal. Both experiments were done by averaging the execution costs for over 1000 key pairs from the same delegation structure. The variation seems to be always greater in the searches with negative answer but we expect such queries to be minority in actual systems.

5 Conclusion

We described and compared several algorithms for authorization decisions from a database of certificates. The algorithms are based on well-known graph-search techniques that have been enhanced to handle joint-delegation certificates. Measurements on generated certificate data were done to compare the efficiency of the algorithms.

The main observations was that it is feasible to make authorization decisions from large delegation networks comprising thousands of keys and certificates. The most efficient algorithm was found to be the two-way search where we first mark keys one or two certificates away from the server with a forward search and then try to locate one of the marked nodes with a backward search. This is a little faster than simple backward search, but more difficult to implement. The two-way search is at its best when there are no joint-delegation certificates while the backward searches handle them particularly well.

References

[1] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proc. 1996 IEEE symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.

[2] Carl M. Ellison, Bill Franz, Butler Lampson, Ron Rivest, Brian M Thomas, and Tatu Ylönen. Simple public key certificate. Internet draft, SPKI Working Group, July 1997.

[3] Ronald L. Rivest and Butler Lampson. SDSI — A simple distributed security infrastucture. Technical report, April 1996.

[4] Tatu Ylönen. Proposal for SPKI certificate formats and semantics. Unpublished manuscript, April 1997.