HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **48**; December 1997

ISSN 0783–5396
ISBN 951–22–3889–6

ON THE STRUCTURE OF DELEGATION NETWORKS

Tuomas Aura
Digital Systems Laboratory
Department of Computer Science and Engineering
Helsinki University of Technology
Otaniemi, FINLAND

Helsinki University of Technology
Department of Computer Science and Engineering
Digital Systems Laboratory
Otaniemi, Otakaari 1
P.O.Box 1100, FIN–02015 HUT, FINLAND

HELSINKI UNIVERSITY OF TECHNOLOGY
DIGITAL SYSTEMS LABORATORY
Series **A:** Research Reports
No. **48**; December 1997

ISSN 0783–5396
ISBN 951–22–3889–6

On the Structure of Delegation Networks

Tuomas Aura

Abstract: In new distributed, key-oriented access control systems access rights are delegated by a freely formed network of certificates. For example, the SPKI public-key infrastructure is being designed for this kind of distributed trust management on the Internet.

We formalize the concept of a delegation network and present a formal semantics for the delegation of access rights with certificates. The certificates can have multiple subjects who must jointly use the authority. Some fundamental properties of the system are proven, alternative techniques for authorization decisions are compared and their equivalence is shown rigorously. In particular, we prove that certificate reduction is a sound and complete decision technique. We also suggest a new type of threshold certificates and prove its properties. The formal model is used to develop efficient algorithms for access control decisions from a database of certificates.

Keywords: certificates, delegation network, access control, formal model of distributed trust management.

Printing: Libella Painopalvelut Oy 1997

Helsinki University of Technology
Department of Computer Science and Engineering
Digital Systems Laboratory
Otaniemi, Otakaari 1
B.O.Box 1100, FIN–02015 HUT, FINLAND

Phone: $\frac{(09)}{+358\ 9}$ 4511
Telex: 125 161 htkk fi
Telefax: +358 9 451 3369
E-mail: lab@saturn.hut.fi

This work has been funded by Helsinki Graduate School in Computer Science and Engineering (HeCSE) and Academy of Finland project number 8309. Also the grants from Tekniikan Edistämissäätiö, Suomen Kulttuurirahaston Rank Xerox Oy:n rahasto, Telecom Finland Oy:n tutkimus- ja koulutussäätiö, and Kaupallisten ja teknisten tieteiden tukisäätiö are gratefully acknowledged.

Contents

1 Introduction **4**

1.1 Outline of the report 4

1.2 Background and related work 5

2 Delegation network **8**

2.1 Definition of delegation network 8

2.2 Set-type authorizations 10

2.3 The authorization problem 11

2.4 Subnetworks 13

3 Tree-based formulation of the authorization problem **19**

3.1 Delegation tree 19

3.2 Trees and the authorization problem 21

4 Certificate reduction **25**

4.1 Definition of certificate reduction 25

4.2 Soundness and completeness of certificate reduction 26

5 Threshold certificates **30**

5.1 (k, n) schemes 30

5.2 Open threshold certificates 30

6 Algorithms for deciding the authorization problem **37**

6.1 Typical delegation network structure 37

6.2 Certificate reduction as an algorithm 38

6.3 Depth-first search forward 39

6.4 Depth-first and breadth-first search backward 41

6.5 Two-way search 43

6.6 Experimental evaluation of the algorithms 45

7 Conclusion **49**

List of Symbols and Abbreviations

Symbol	Definition	Page
$Arcs$	set of arcs in a tree	21
$auth$	function binding authorizations to certificates	8
$Auths$	set of authorizations i.e. access rights	8
$authorizes$	three-place relation describing who is authorized to what	11, 31
β	branching factor of the network	43
BFS	Breadth-First Search	
c	certificate	
$Certs$	set of certificates	8
DFS	Depth-First Search	
DN	delegation network	8
DT	delegation tree	21
$Flow$	relation from issuers to certificates to subjects	8
h	homomorfism from tree to delegation network	19
id	certificate group identifier	
Ids	set of certificate group identifiers	31
$issuers$	issuers of a set of certificates	33
k	key	
(k, n)	threshold scheme	30
$Keys$	set of cryptographic public–private key pairs	8
n	node in a tree	
$Nodes$	set of nodes in a tree	21
o	operation	
$OAuths$	open-threshold-type authorizations	31
$Open$	transformation to open-threshold-type	31
Ops	set of operations i.e. atomic access rights	10
SDSI	Simple Distributed Security Infrastructure	6
SPKI	Simple Public Key Infrastructure	6
$\mathbb{Z}^+$	positive integers	

1 Introduction

New key-oriented access control systems offer a fully distributed alternative to traditional hierarchical or centralized, identity-oriented schemes. In the new systems, access rights are bound to a key, not to the identity of the owner of the key. They are delegated from key to key with chains of signed certificates. These certificates form a network between the keys, where the amount of trust between each two keys can be exactly specified. This way, local authorities are free to establish trust relations without the need for a global hierarchy of trusted officials.

The goal of this paper is to present an abstract model for the networks of delegation formed by public-key certificates between keys. We formalize the concept of a *delegation network* and present a formal semantics for delegation. The model is used for proving the equivalence of different methods for access control decisions. In particular, we show that the certificate reduction technique of [15] is sound and complete with respect to our definition of authorization. Theoretical treatment of the topic allows us to focus on the essential features of the systems instead of lengthy technical specifications. This makes it possible to develop efficient algorithms for access control decisions from a database of certificates. We also show that joint delegation certificates of [15] can be slightly generalized while simplifying the implementation.

1.1 Outline of the report

We begin with a brief overview of the history and development of access control models in Sec. 1.2.

The concept of delegation network and the authorization problem are defined in Sec. 2. This section also discusses subnetworks and presents a fundamental theorem on the existence finite ones.

Sec. 3 shows how delegation can be visualized as trees. This is helpful in proving theorems and in development of algorithms.

Certificate reduction as a technique for deciding the authorization problem is introduced and its soundness and completeness is proven in Sec. 4. This is the most important result of the report.

Our theory allows all certificates to be joint-delegation type, i.e. to have several subjects whose co-operation is needed for using the delegated rights. Sec. 5 discusses threshold certificates where only a certain threshold number of subjects is required to co-operate. A generalized type of threshold

certificates is described and its security properties are proven.

Sec. 6 contains algorithms for deciding the authorization problem from a database of certificates. The structure of a typical delegation network is discussed. The model is based on conceptual analysis since examples of implementations are not yet available. An efficient two-way search algorithm for large sets of certificates is presented. The efficiency of alternative algorithms is compared based on the model of typical delegation network structure. The expected performance is compared to simulations on generated data.

Sec. 7 concludes the report and makes some remarks on possible directions of future research.

1.2 Background and related work

Traditionally, access control decisions in a system have been made by a central authority called reference monitor. The idea is that access requests go through a trusted system component that decides if they should be allowed. The authority can, for example, be an operating system or a database manager.

The reference monitor concept cannot easily be adapted [20, 7, 25] to the highly distributed systems built around today's data communications networks [16, 26]. In the network, a virtually unlimited number of local authorities can set up and administer access to their own resources. Furthermore, from each host's viewpoint, the network can be divided into areas of more or less trusted and untrusted hosts, e.g. separated by firewalls [12].

The reference monitor usually follows some fixed access control policy. The most common types of policies have been based on labeling of the subjects and objects with multi-level labels. Higher level data is more sensitive and higher level entities have more access rights. Well-known models of multi-level security are the Bell-LaPadula [4] and Biba [5] models.

Multi-level security is suitable for centralized multi-user computers in a high-security environment such as military organizations. It does not necessarily satisfy the needs of commercial environment or those of private persons. In a commercial environment, the separation of duties between trusted entities is often just as important as protection against outsiders. The Clark–Wilson [13] and Chinese Wall [9] models aim to do this from different angles of view.

Another problem with multi-level security is that the concept of a single global security policy does not scale well to computer networks. Since the criteria for granting or denying access depend on the provider of the service,

the policies in a network environment can be as diverse as the interests of the networked community.

The PGP [31, 27] approach does away with authorities. Instead, a web of trust is allowed to anarchically develop between individual persons on the net. The central objective is trustworthy certification of identities. However, because of the intransitivity of complete trust, the web-of-trust concept is mostly used for managing personal relations.

Some systems like the X.509 authentication hierarchy [10, 17] and Kerberos [18] try to scale the centralized, identity-based approach to open networks. In X.509, trustworthiness radiates from a central trusted entity to lower level authorities. It has the obvious problem that, in the end, everyone has to trust all the officials appointed by the global central authority. In the Kerberos authentication service, the goals are more modest and it has worked well in local network domains. Problems arise when the system should be used between arbitrary nodes on a large network. Some serious attempt are being made to combine the ideas from PGP web of trust and X.509 hierarchy into a web of local hierarchies [11, 30]. It is too early to see how successful this will be in practice.

Both the anarchical web of trust and the more centralized and hierarchical systems have had their main emphasis of verifying the identities of individuals. The certification authorities must be completely trusted with respect to all activities for which the certified keys are used. This may be why a right balance between centralization and free formation of trust relations has not been found and no general solution exists to access control problems on the networks. The new kind of distributed, key-oriented authentication infrastructures address the problems by replacing identities with cryptographic keys owned by individuals and computer systems. They allow free creation of local and global authorities and trust relationships between them. Also, the delegated rights, i.e. the level of trust, can be precisely specified in the certificates.

The three most prominent proposals for distributed trust management are SPKI certificates [15] by Ellison et al., SDSI public key infrastructure [23] by Rivest and Lampson, and PolicyMaker local security policy database [8] by Blaze et al. SDSI replaces globally unique names of entities with linked local name spaces [1]. SPKI is a standard proposal for certificates whose purpose is to delegate access rights rather than to certify identity. PolicyMaker is a general database for managing access control policies. In the development of our theory, we have most often referred to the SPKI specification.

A lot of work has been done on modelling the structure and behavior of systems under the control of a single reference monitor. For example, the

take–grant model can be used to characterize different access control policies [28, 6]. There is, however, very little literature on the new key-oriented systems. Especially theoretical treatments have not been published. This is the gap we are trying to fill.

2 Delegation network

We start by defining a structure called *delegation network* in Sec. 2.1. It consists of keys and certificates delegating authorizations between the keys. The authorizations are rights to perform sets of operations. This is detailed in Sec. 2.2. In Sec. 2.3 we continue by formulating the authorization problem, i.e. the question of who is authorized to which operations, in terms of the delegation networks. Subnetworks and a fundamental result on their existence is presented in Sec. 2.4.

2.1 Definition of delegation network

We define a delegation network as a directed bipartite graph. The partitions of nodes are called keys (the set *Keys*) and certificates (the set *Certs*). The certificates are annotated with authorizations (the set *Auths*). The directions of arcs (the *Flow* relation) point from the issuer key to the certificate and from the certificate to the subject keys. With the certificate, the issuer delegates to the subject(s) the right to (jointly) request some operations to be executed. In our level of abstraction, the keys are primitive data items in the sense that we will not give any structure to the set. The relations between keys are determined by their connections to the certificates. This way, we abstract away the cryptography that will make keys and certificates work in implementations. The authenticity of the certificates must have been checked by verifying signature on them at the time the certificates were entered into the database. The set of authorizations, on the other hand, will be given a structure in Sec. 2.2.

Definition 1 (delegation network) *A* delegation network *is a 5-tuple* $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ *such that*

1. $Keys$ *is a set called* keys,
2. $Certs$ *is a set called* certificates,
3. $Auths$ *is a set called* authorizations,
4. $Flow \subseteq Keys \times Certs \cup Certs \times Keys$ *is called a* flow relation,
5. *for each* $c \in Certs$, *there is a unique key* $k \in Keys$ *such that* $\langle k, c \rangle \in Flow$. *This key is called the* issuer *of* c.
6. *for each* $c \in Certs$, *there is at least one and at most a finite number of keys* k *such that* $\langle c, k \rangle \in Flow$. *These keys are called the* subjects *of* c.

7. *auth* : *Certs* $\longrightarrow$ *Auths maps certificates into authorizations.*

According to the definition, a certificate is connected to two or more other keys. For exactly one of these keys, the arc is directed towards the certificate. This key is the issuer, i.e. signer, of the certificate. The other keys, subjects, are the keys to whom the certificate has been given. The function *auth* attaches to each certificate the access rights delegated with it.

We limit the number of subjects for each certificate to finite although the number of certificates in the network can be infinite. This makes sense because representing an infinite set of cryptographic keys in one certificate does not seem implementable but the number of certificates retrievable from, for example, a computer network can be unlimited.

The certificates could also be defined as a relation between keys. We have chosen the graph approach, because it makes the theory more visual and we will draw ideas for decision algorithms from graph theory. It should be noted that if all certificates have only a single subject, the nodes representing them have only one incoming arc and one leaving arc. In that case, the certificates can be pictured as annotations on arcs between the keys.

Note that we allow delegation networks to have cycles, i.e. a key can directly or indirectly delegate access rights to itself. This kind of cyclic delegation naturally will not give the key any new rights. It merely means that the alternative paths of delegation form loops. For simplicity, we also do not want to disallow direct delegation to self although it is never useful in practice. We will, however, show that in some situations it suffices to look at parts of delegation networks with no cycles. Therefore we give the following definition.

Definition 2 (acyclic) *A delegation network with flow relation Flow is* acyclic *iff the network has are no* cycle*, i.e. looping chains of certificates*

$$\langle k_1, c_1\rangle, \langle c_1, k_2\rangle, \langle k_2, c_2\rangle, \langle c_2, k_3\rangle, \ldots, \langle c_{n-1}, k_n\rangle \in \mathit{Flow}$$

where $k_1 = k_n$.

Fig. 1 shows an example of a delegation network. On the certificates, we have marked the access rights delegated by them. Only one certificate has more than one subject. The network has a cycle although no access rights are actually delegated all the way around the cycle.

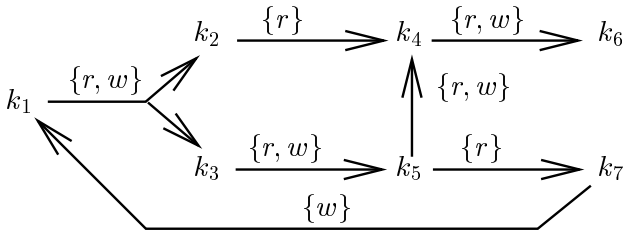

Figure 1: A delegation network

2.2 Set-type authorizations

The *auth* function specifies the access rights delegated with a certificate. The structure of the authorizations depends on what kind of access takes place in the system.

Often, authorizations are a sets of operations that the subject of the certificate is allowed to request. In that case, the result of a series of delegations is given by the intersection of the operation sets allowed in the delegations and the result of obtaining access rights from several sources is given by the union of the operation sets.

Definition 3 (set-type authorizations) Set type authorizations *are formed by a lattice of subsets of a set of operations.*

Thus, the authorizations are sets of operations, $Auths \subseteq P(Ops)$ (the power set of Ops) for some set Ops. The word lattice in this context means that the union and the intersection of any two authorizations must also be authorizations.

If $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ is a delegation network, the set $Ops = \cup Auths$ is called the *operations of* DN.

The set-type authorizations have the advantage that the right to perform each operation can be considered separately. There is no need to define special operations for combining the rights obtained by a single key from several certificate paths. Instead, the certificates can be presented together to demonstrate the right to the union of the access rights delegated by each of them. This makes the implementation of the system straightforward. It would be possible to define authorizations with more complex structure, for example, by allowing arbitrary functions for combining them as in [8].

2.3 The authorization problem

We will now define how the access rights are transfered from key to key in a delegation network. This is the most straightforward way to define the semantics of the authentication networks since it raises directly from the intuitive meaning of the certificates. Access rights are transfered to the set of subjects who all must delegate the right to the same key, possibly via other keys. When there is only one subject, that subject can alone use or delegate the rights. Of course, every key completely trusts itself.

Definition 4 (*authorizes* relation) *Let* $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ *be a delegation network where the authorizations are set-type. Denote by Ops the operations of DN. The relation* $authorizes_{DN} \subseteq Keys \times Keys \times Ops$ *is the smallest three-place relation such that*

1. *if* $k \in Keys$ *and* $o \in Ops$*, then* $\langle k, k, o \rangle \in authorizes_{DN}$*, and*
2. *if* $\langle k_1, c \rangle \in Flow$ *and* $\langle k, k_2, o \rangle \in authorizes_{DN}$ *for all* k *such that* $\langle c, k \rangle \in Flow$*, then* $\langle k_1, k_2, o \rangle \in authorizes_{DN}$.

Lemma 5 *With the assumptions of Def. 4, there is a unique smallest relation (with respect to set inclusion) satisfying the two rules in the definition.*

Proof Assume that $authorizes_i$ are two or more differing minimal relations satisfying the two rules in Def. 4. The intersection of the relations $authorizes = \cap authorizes_i$ is smaller than either of the two relations. Furthermore, the intersection satisfies the two conditions: (1) $\langle k, k, o \rangle$ must be included in all the relations, so it also is in the intersection. (2) If $\langle k_1, c \rangle \in Flow$, and $\langle k, k_2 \rangle \in authorizes$ for all k such that $\langle c, k \rangle \in Flow$, then also $\langle k, k_2 \rangle \in authorizes_i$ for the same keys k for all i. Therefore, $\langle k_1, k_2, o \rangle \in authorizes_i$. Thus, $\langle k_1, k_2, o \rangle$ is in $authorizes$ and Rule 2 is satisfied. Since the intersection $authorizes$ is smaller than all of the relations and also satisfies the two rules, the relations $authorizes_i$ cannot be minimal. In conclusion, the assumption of having two different minimal relations is wrong and there is a unique smallest relation $authorizes_{DN}$. □

Note that the definition does not refer to the graph terminology at all. In Sec. 3 we will give an equivalent formulation based on trees in the graph.

Often, we will write $authorizes_{DN}(k_1, k_2, o)$ in predicate notation to denote $\langle k_1, k_2, o \rangle \in authorizes_{DN}$. If *ops* is a set of operations, and we have $authorizes_{DN}(k_1, k_2, o)$ for all $o \in ops$, we write $authorizes_{DN}(k_1, k_2, ops)$.

When $authorizes_{DN}(k_1, k_2, o)$ is true, we say that key k_1 delegates authorization for operation o to key k_2 in DN. This is the central question to be queried from a database of certificates, called the *authorization problem.*

The authorization problem
In a database of certificates, does a key k_1 delegate authorization for operation o to another key k_2, i.e. is $authorizes_{DN}(k_1, k_2, o)$ true in the delegation network?

For example, in the network of Fig. 1, $authorizes(k_1, k_6, r)$ is true, but $authorizes(k_1, k_6, w)$ and $authorizes(k_1, k_7, r)$ are not true because the delegation path through k_2 is missing.

Usually, the first key in the chain of delegation should be a key belonging to the server providing the service for which authorization is being delegated. This way, the server who naturally trusts its own public key can verify from the set of certificates that the client key has the right to request the service.

The idea of minimality in Def. 4 is that all tuples in the relation *authorizes* should have an explicit reason for being there. It is a straightforward consequence of the minimality that in order for a triple $\langle k_1, k_2, o\rangle$ to be in the relation *authorizes*, one of the Rules 1 and 2 must be the reason. This is formally stated in the following lemma.

Lemma 6 *Let DN be a delegation network. For all keys k_1 such that $\langle k_1, k_2, o\rangle \in authorizes_{DN}$, at least one of the following holds:*

1. *$k_1 = k_2$, or*
2. *there exist $c \in Certs$ such that $\langle k_1, c\rangle \in Flow$ and $\langle k, k_2, o\rangle \in authorizes_{DN}$ for all k such that $\langle c, k\rangle \in Flow$.*

Proof The theorem is proven by contradiction. Assume that there is an element $\langle k_1, k_2, o\rangle \in authorizes_{DN}$ for which neither of the conditions 1–2 of Lemma 6 holds. The relation $authorizes' = authorizes_{DN} \setminus \{\langle k_1, k_2, o\rangle\}$ is smaller than $authorizes_{DN}$. Furthermore, neither of the conditions in Def. 4 requires $\langle k_1, k_2, o\rangle$ to be a member of the relation, and removing an element from the relation cannot make the conditions true for any other element. Thus, the smaller relation still satisfies the requirements of Def. 4. Therefore, the assumption cannot be true. □

In addition, the minimality of *authorizes* means that looping or infinite chains of certificates do not add to the relation. A consequence is that in

order to have effect, any path of delegation must end in a certificate that has only a single subject. This is stated formally in the next theorem. Although the theorem does not depend on any concepts other than those presented so far and could thus be proven here, the proof is delayed till the end of Sec. 2.4 where we have some technically convenient results at hand.

Theorem 7 *Let DN be a delegation network such that $authorizes_{DN}(k_1, k_2, o)$ for two keys $k_1 \neq k_2$. There is a certificate c in DN whose only subject is k_2.*

2.4 Subnetworks

Even if the delegation network is very large or infinite in size, decisions to grant access are based on finite subsets of certificates. For this purpose, we define the concept of a *subnetwork*. A subnetwork is a part of a delegation network that has some of the keys and certificates of the original network so that all the keys connected to the remaining certificates are also retained.

Definition 8 (subnetwork) *Let $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ be a delegation network. $DN' = \langle Keys', Certs', Auths', Flow', auth' \rangle$ is a* subnetwork *of DN iff $Keys' \subseteq Keys$, $Certs' \subseteq Certs$, $Auths' \subseteq Auths$, and $Flow'$ and $auth'$ are restrictions of $Flow$ and $auth$, respectively, to $Keys'$ and $Certs'$, and the following condition is satisfied: $c \in Certs' \wedge (\langle k, c \rangle \in Flow \vee \langle c, k \rangle \in Flow) \Rightarrow k \in Keys'$.*

If DN' is a subnetwork of DN, we say that DN is a *supernetwork* of DN'.

The authorization relation in a subnetwork is naturally a subset of the relation for a supernetwork. This is because the rules in Def. 4 cannot be disabled by adding new keys and certificates to the delegation network.

Theorem 9 *Let DN be a delegation network with set-type authorizations and DN' its subnetwork. In that case, $authorizes_{DN'} \subseteq authorizes_{DN}$.*

Proof Let $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ be a delegation network and $DN' = \langle Keys', Certs', Auths', Flow', auth' \rangle$ its subnetwork . By the definition of subnetwork (Def. 8), $Keys' \subseteq Keys$, $Certs' \subseteq Certs$ and $Flow' \subseteq Flow$. Def. 4 defines $authorizes_{DN'}$ as the smallest relation including all tuples that satisfy certain two conditions (the left sides of the two rules in the definition). These conditions are monotonic in the way that they cannot be made false by adding new items into the sets $Keys'$, $Certs'$ and $Flow'$. Therefore, if $\langle k_1, k_2, o \rangle \in authorizes_{DN'}$ by the rules, also $\langle k_1, k_2, o \rangle \in$

authorizes$_{DN}$ for any delegation network DN with equal or larger sets of keys, certificates and flow relation. □

The next theorem is the basis for most of the following theory and for development of decision algorithms. It shows that we only need to consider finite subsets of certificates when deciding if the relation *authorizes*$_{DN}$ holds for a pair of keys. The proof is particularly interesting because its first part contains a construction of the relation *authorizes*$_{DN}$.

Theorem 10 *Let DN be a delegation network where authorizes$_{DN}(k_1, k_2, o)$. DN has a finite acyclic subnetwork $DN' = \langle Keys', Certs', Auths, Flow', auth' \rangle$ where authorizes$_{DN'}(k_1, k_2, o)$ and, furthermore,*

1. *authorizes$_{DN'}(k, k_2, o)$ is true for all the keys $k \in Keys'$,*
2. *k_2 is the only key in $Keys'$ that is not an issuer of any certificate in $Certs'$, and*
3. *$o \in auth'(c)$ for all $c \in Certs'$.*

Proof (including construction of *authorizes*) In the first part of the proof we follow the flow relation from the subject keys (in particular from k_2) towards issuers and get a subset of certificates where the maximum length of delegation paths is bounded. In the second part, we follow the flow from k_1 towards k_2 and remove all but one of the alternative delegation paths. The result is a finite subnetwork with the desired properties.

We first consider an arbitrary operation o and a subject key k and see which keys delegate the right for the operation o to the key k. These keys and the certificates delegating the right to o will be collected in indexed sets by increasing length of delegation paths to k. As an initial step, define the sets

$$\begin{aligned} Certs_0^{k,o} &= \emptyset, \\ Keys_0^{k,o} &= \{k\}, \\ A_0^{k,o} &= \{\langle k, k, o\rangle\}. \end{aligned}$$

Then, for $i = 1, 2, \ldots$, define

$$\begin{aligned} Certs_i^{k,o} &= \{c \mid o \in auth(c) \ \wedge \\ &\quad (\forall k' : (\langle c, k'\rangle \in Flow \Rightarrow k' \in \cup_{j=0}^{i} Keys_j^{k,o}) \\ &\quad\quad \wedge (\langle k', c\rangle \in Flow \Rightarrow k' \notin Keys_{i-1}^{k,o}))\}, \\ Keys_i^{k,o} &= \{k\} \cup \{k' \mid c \in Certs_i^{k,o} \wedge \langle k', c\rangle \in Flow\}, \\ A_i^{k,o} &= \{\langle k', k, o\rangle \mid k' \in Keys_i^{k,o}\}. \end{aligned}$$

Corresponding cumulative collections of keys and certificates are

$$Certs_l^{*k,o} = \cup_{i=o}^{l} Certs_i^{k,o},$$
$$Keys_l^{*k,o} = \cup_{i=o}^{l} Keys_i^{k,o}.$$

We show by induction that $Keys_i^{*k,o}$ and $Certs_i^{*k,o}$ cannot form infinite paths of keys and certificates. Basis step: The maximum path length of *Flow* in $Keys_0^{*k,o} \cup Certs_0^{*k,o}$ is 0. This is because all paths contain only a single key. Induction step: If the maximum path length of *Flow* in $Keys_i^{*k,o} \cup Certs_i^{*k,o}$ is finite, then in $Keys_{i+1}^{*k,o} \cup Certs_{i+1}^{*k,o}$ it is extended at most by 2. Infinite paths cannot be formed for two reasons. Firstly, the extensions to paths lead to new keys that were not in the previous set. Hence, loops cannot be formed with earlier keys and certificates. The extensions only increase length of existing paths. Secondly, the extensions themselves cannot connect to each other forming loops or infinite paths because the subjects of the new certificates are all in the earlier sets. Only the issuer is in the new set. By induction, the maximum path length for *Flow* in $Keys_i^{*k,o} \cup Certs_i^{*k,o}$ is finite meaning also that loops do not exist for any $i \geq 0$.

Moreover, $\langle k', k, o\rangle \in A_i^{k,o}$ for all $k' \in Keys_i^{k,o}$ for all $i = 0, 1, 2, \ldots$, and $A_i^{k,o} \subseteq authorizes_{DN}$. In the basis step this follows from Rule 1 of Def. 4 and later from Rule 2 of the same definition.

We now construct $authorizes_{DN}$ as a union of the sets $A_i^{k,o}$. Denote the set of operations of DN by *Ops* and let

$$A = \cup_{o \in Ops} \cup_{k \in Keys} \cup_{i=0}^{\infty} A_i^{k,o}.$$

Based on the results of the previous paragraph, $A \subseteq authorizes_{DN}$. Also, A is closed in DN with respect to the two rules of Def. 4. Rule 1 is satisfied because $\cup_{o \in Ops} \cup_{k \in Keys} A_0^{k,o} \subseteq A$. For Rule 2, consider any $\langle k_1', c\rangle \in Flow$ for which $\langle c, k\rangle \in Flow$ implies $\langle k, k_2', o\rangle \in A$. Since the number of subjects k of c is finite, there is some finite i so that $k \in Keys_i^{k_2',o}$ for all the subjects k. If $k_1' \in Keys_i^{k_2',o}$ then $\langle k_1', k_2', o\rangle \in A_i^{k_2',o}$. If $k_1' \notin Keys_i^{k_2',o}$ it follows from our construction of the sets that $k_1' \in Keys_{i+1}^{k_2',o}$ and $\langle k_1', k_2', o\rangle \in A_{i+1}^{k_2',o}$. In both cases, $\langle k_1', k_2', o\rangle \in A$. Hence, A fulfills the two closure rules of $authorizes_{DN}$. Since we also know that $A \subseteq authorizes_{DN}$, the minimality of $authorizes_{DN}$ implies $A = authorizes_{DN}$.

Note that the issuers and subjects of all certificates of $Certs_i^{*k,o}$ are in $Keys_i^{*k,o}$. Moreover, the sets above are constructed in such a way that for all $k' \in Keys_i^{k,o}$ except for k, there is a certificate issued by k in $Certs_i^{k,o}$ and the subjects of the certificate are all in $Keys_{i-1}^{*k,o}$. Thus, for all i, k is the only

key in $Keys_i^{*k,o}$ that is not an issuer of any certificate in $Certs_i^{*k,o}$, and all certificates of $Certs_i^{*k,o}$ allow operation o. These properties will be retained in the further reduced sets of certificates in the second part of the proof.

Since $\langle k_1, k_2, o\rangle \in A$, we have $k_1 \in Keys_j^{k_2,o}$ and $\langle k_1, k_2, o\rangle \in A_j^{k_2,o}$ for some $j \in \{0, 1, 2, \dots\}$. This j is the maximum length of delegation paths that need to be considered for $authorizes_{DN}(k_1, k_2, o)$ to be found true.

We now get to the second part of the proof. The subnetwork DN' will be formed by following the delegation paths in $Keys_j^{*k_2,o} \cup Certs_j^{*k_2,o}$ from the key k_1 towards the subjects. On the way, we select one of all alternative ways in which the rights reach the key k_2. As the path lengths are finitely bounded, the chosen paths will terminate at k_2 after a finite number of steps.

Let $Keys_j = \{k_1\}$ and let $Certs_j = \{c\}$ be a singleton containing (an arbitrarily chosen) one of the certificates in $Certs_j^{k_2,o}$ such that $\langle k_1, c\rangle \in Flow$. According to the definition of $Keys_j^{k,o}$, at least one such c must exist. Otherwise, k_1 would not be in $Keys_j^{k,o}$.

For $i = j-1, j-2, \dots, 1, 0$ define:

$$Keys_i = \{k \mid \langle c, k\rangle \in Flow \wedge c \in Certs_{i+1}\}.$$

Also, build the set $Certs_i$ by choosing for each $k \in Keys_i$ one certificate $c \in Certs_i^{k_2,o}$ such that $\langle k, c\rangle \in Flow$. Again, such a c must exist because otherwise k would not be in $Keys_i^{k_2,o}$.

The finite and acyclic subnetwork DN' is constructed as follows. Denote $Keys' = \cup_{i=0}^{j} Keys_j$ and $Certs' = \cup_{i=0}^{j} Certs_j$. The delegation network $DN' = \langle Keys', Certs', Auths, Flow', auth'\rangle$ where $Flow'$ and $auth'$ are restrictions of $Flow$ and $auth$ (respectively) to $Keys' \cup Certs'$, is a subnetwork of DN because the issuer and the subjects of each certificate of $Certs_i$ are in $Keys_i \cup Keys_{i-1}$ and thus in $Keys'$.

We now show that DN' is acyclic and finite. Since the paths of $Flow$ in $Keys' \cup Certs'$ are a subset of the paths in $Keys_j^{*k,o} \cup Certs_j^{*k,o}$, the paths lengths are bounded by a finite number j also in DN'. Hence, the paths are acyclic. The number of keys and certificates in $Keys_j \cup Certs_j$ is finite (actually there is one key and one certificate). For each index $i = j-1, j-2, \dots$, the number of keys and certificates in $Keys_i \cup Certs_i$ remains finite because the number of subjects for each certificate is finite. Since the lengths of the paths are finite, the total number of keys and certificates chosen to DN' is finite. Thus, DN' is acyclic and finite, as suggested in the theorem.

On each level of the construction, $i = j-1, j-2, \dots, 1, 0$, $Keys_i$ and $Certs_i$ are non-empty because of the way in which $Keys_i^{k,o}$ and $Certs_i^{k_2,o}$

were constructed guarantees that all certificates of $Certs_i^{k_2,o}$ have subjects in $Keys_{i-1}^{*k_2,o}$. Thus, $Keys_0 = \{k_2\} \subseteq Keys'$.

We show by induction that $authorizes_{DN'}(k, k_2, o)$ for all $k \in Keys'$. The basis step: for the single key $k_2 \in Keys_0$, $authorizes_{DN'}(k_2, k_2, o)$ follows from Rule 1 of Def. 4. The induction step: assume that $authorizes_{DN'}(k, k_2, o)$ for all $k \in Keys_i$. The sets $Certs_{i+1}^{k_2,o}$ and $Keys_{i+1}^{k_2,o}$ were specifically constructed so that all $k \in Keys_{i+1}^{k_2,o}$ issue a certificate in $Certs_{i+1}^{k_2,o}$ and all $c \in Certs_{i+1}^{k_2,o}$ have subjects in $Keys_i^{*k_2,o}$. When we above chose some of these keys to $Keys_i$ and $Keys_{i+1}$ and some certificates to $Certs_{i+1}$, this was done in such a way that all keys of $Keys_{i+1}$ still issue a certificate in $Certs_{i+1}$ and all subjects of this certificate are still in $Keys_i$. By Rule 2 of Def. 4 it follows that $authorizes_{DN'}(k, k_2, o)$ for all $k \in Keys_{i+1}$. By induction, $authorizes_{DN'}(k, k_2, o)$ for all $k \in Keys'$. This suffices to show Claim 1 of the theorem. Naturally also $authorizes_{DN'}(k_1, k_2, o)$.

The construction guarantees directly that keys other than k_2 in $Keys'$ are issuers of certificates in $Certs'$. Key k_2 cannot be the issuer of any certificate, because the issuers of new certificates to $Certs_i^{k_2,o}$ are required not to be in the previous sets $Keys_{i-1}^{*k_2,o}$, and $k_2 \in Keys_i^{*k_2,o}$ for all $i = 0, 1, 2, \ldots$. Thus, Claim 2 holds for DN'.

Finally, only certificates c for which $o \in auth'(c)$ were chosen to $Certs_j^{*k_2,o}$ and consequently to $Certs'$. This concludes the proof of Claim 3 and of the entire Theorem 10. □

The above theorem is consequence of the requirement for the set of subjects of a single certificate to be finite. If we would allow a certificate to have an infinite number of subjects, the finiteness of the subnetwork in the above theorem would not hold. It is interesting to note that the absence of infinite length paths could still be proven. Fig. 2 illustrates the peculiar situation where a certificate has an infinite number of subjects and all paths have finite length but the path lengths do not have any upper bound. We are, however, interested in delegation that depends only on finite number of certificates and, thus, can be decided algorithmically.

One further detail to note is that the reflexive transitive closure of the flow relation in an asymmetric delegation network is a partial order on the keys and certificates.

Theorem 11 *Let DN be a finite and acyclic delegation network with flow relation $Flow$. The reflexive and transitive closure of $Flow$ is a partial order on the keys and certificates.*

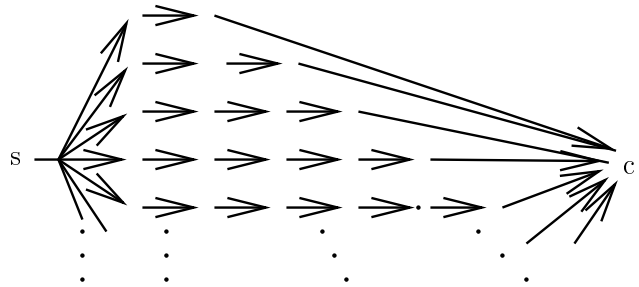

Figure 2: A certificate with an infinite number of subjects

Proof The reflexive and transitive closure of an acyclic graph is reflexive, transitive and antisymmetric, and thus, a partial order. □

Next we will prove Theorem 7. The theorem itself is a consequence of the requirement for the *authorizes* relation to be minimal and it does not involve subnetworks is any way. Nevertheless, we give the proof at this point of discussion because it is easier to present with the help of Theorem 10.

Proof of Theorem 7 Let DN be a delegation network such that $authorizes_{DN}(k_1, k_2, o)$ for two keys $k_1 \neq k_2$. Assume that all certificates of DN that have k_2 as a subject also have at least one other subject.

According to Theorem 10, DN has a finite, acyclic subnetwork $DN' = \langle Keys', Certs', Auths', Flow', auth' \rangle$, where also $authorizes_{DN'}(k_1, k_2, o)$. A finite and acyclic subnetwork has no infinite chains of keys and certificates such that $\langle k'_1, c'_1 \rangle, \langle c'_1, k'_2 \rangle, \langle k'_2, c'_2 \rangle, \langle c'_2, k'_3 \rangle, \ldots \in Flow$.

Since the certificates in DN' are a subset of those in DN, and their subjects are preserved, it follows that all certificates in DN' that have k_2 as a subject, also have at least one other subject.

We choose $k'_1 = k_1$. Since $authorizes_{DN}(k_1, k_2, o)$ and $k_1 \neq k_2$, Lemma 6 says that there must exist a certificate c'_1 issued by k_1 for all of whose subjects k, $authorizes_{DN}(k, k_2, o)$. By our assumption, one of the subjects is not equal to k_2. We choose this subject as k'_2. We already have $authorizes_{DN}(k'_2, k_2, o)$ and $k'_2 \neq k_2$ so we take k'_2 as the next starting point and find a certificate c'_2 and subject k'_3. Continuing this way, we get an infinite chain of keys and certificates where a subject of the precious certificate always issues the next certificate. But such chains cannot exist in an acyclic network. Thus, the assumption is false and there is a certificate in DN' and in DN whose only subject is k_2. □

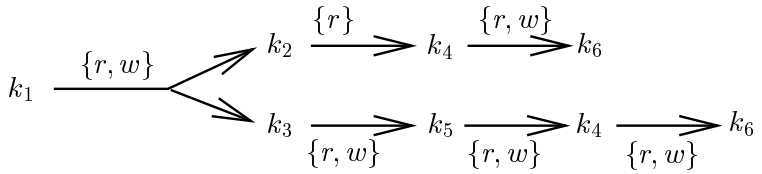

Figure 3: A delegation tree

3 Tree-based formulation of the authorization problem

In this section, we will reformulate the authorization problem with graph terminology. If a key k_1 delegates access rights to another key k_2, a tree of keys and certificates can be formed such that k_1 is at the root of the tree and all branches end to k_2. The tree-based representation of delegation will help us to visualize the theory and to make proofs more intuitive (see Sec. 4), and it has played a key role in development of graph-search algorithms for delegation decisions in Sec. 6.

We formally define the tree in Sec. 3.1 and show in Sec. 3.2 that such a tree exists if and only if the *authorizes* relation holds.

3.1 Delegation tree

Figure 3 shows how part of the delegation network of Fig. 1 can be unfolded into a tree. This tree shows how the right to operation r is delegated from k_1 to k_6.

Formally, a tree $\langle \mathit{Nodes}, \mathit{Arcs} \rangle$ is an acyclic directed graph formed by a set of nodes *Nodes* and arcs $\mathit{Arcs} \subseteq \mathit{Nodes} \times \mathit{Nodes}$ connecting them. If $\langle n, n' \rangle \in \mathit{Arcs}$, the node n is called the *parent* of n' and n' is called a *child* of n. There is a unique node, called *root node*, with no parent. All other nodes have a unique parent. The nodes with no children are called *leaf nodes*. A tree is *finite* if the number of nodes and arcs is finite. The *depth* of a tree is the maximum path length from a leaf to the root.

For a set of nodes *Nodes* and a function h, we denote $h(\mathit{Nodes}) = \{h(n) \,|\, n \in \mathit{Nodes}\}$.

An annotation of the nodes of a tree with keys and certificates of the network, can be formalized as a homomorfism from the tree to the delegation network.

Definition 12 (homomorfism from tree to delegation network) *Let*

$DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ be a delegation network and $T = \langle Nodes, Arcs \rangle$ a tree. A function $h : Nodes \longrightarrow Keys \cup Certs$ is a homomorfism *from DT to DN iff for all nodes $n, n' \in Nodes$ the following hold:*

1. *if $\langle n, n' \rangle \in Arcs$ then $\langle h(n), h(n') \rangle \in Flow$,*
2. *if $h(n) \in Certs$, there is exactly one node n' such that $\langle h(n'), h(n) \rangle \in Flow$,*
3. *if $h(n) \in Certs$, then h is a bijection from the nodes such that $\langle n, n' \rangle \in Arcs$ to the keys such that $\langle h(n), k \rangle \in Flow$.*

According to the definition, h is simply a homomorfism from a tree to a bipartite graph where the local structure around one of the partitions, certificates, is preserved. We require a node corresponding to a certificate to have a parent corresponding to the issuer and children with 1-1 correspondence to the subjects of the certificate. (The latter requirement is not essential for our theory but it makes the concept of homomorfism more intuitive.)

In Conditions 2 and 3 of the above definition, we implicitly assume that if a node corresponds to a certificate, its parent and child nodes correspond keys. This follows from Condition 1 and the bipartite structure of the delegation network. The converse also holds, i.e. parents and children of nodes corresponding to keys correspond to certificates. Moreover, the root and the leaf nodes of the tree map into keys. This is because every certificate must have an issuer and a subject and they are preserved in the tree.

Lemma 13 *Let h be a homomorfism from a tree DT to a delegation network $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$. If a node n is mapped by h into a certificate, its parent and children are mapped into keys. Also, if a node is mapped into a key, its parent and children (if any exist) are mapped into certificates.*

Proof Assume that nodes n and n' map both into keys or both into certificates and $\langle n, n' \rangle \in Arcs$. Condition 1 in Def. 12 states that $\langle h(n), h(n') \rangle \in Flow$. Thus, two keys or two certificates are connected in the delegation network, but this is not possible according to Def. 1 where the flow relation only connects keys to certificates and certificates to keys. □

The homomorfism always maps the boundary of a tree, i.e. its root and leafs, into keys. This is because we want the issuer and subjects of all certificates to be precisely copied from the delegation network to the tree.

Lemma 14 *Let h be a homomorfism from a tree DT to a delegation network $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$. h maps the root node and all the leaf nodes of DT into $Keys$.*

Proof Def. 1 requires every certificate to have at least one subject key. Condition 3 of Def. 12 requires the nodes of the tree mapping into certificates to have children corresponding to all the subjects of the certificate. Thus, a certificate node always has children and it cannot be a leaf node.

Similarly, Condition 2 of the same definition says that a certificate node always has a parent node mapping into the issuer key of the certificate. Thus, a node corresponding to a certificate is never the root node. □

A delegation tree is simply a tree together with a homomorfism into a delegation network.

Definition 15 (delegation tree) *Let DN be a delegation network. We say that $DT = \langle Nodes, Arcs, h \rangle$ is a* delegation tree *in DN iff $\langle Nodes, Arcs \rangle$ is a finite tree and h is a homomorfism from $\langle Nodes, Arcs \rangle$ to DN.*

When certificates have only one subject, delegation trees reduce into simple paths in the graph. When there are more subjects, the paths branch into trees.

3.2 Trees and the authorization problem

We will show that the finite delegation trees suffice to completely characterize the delegation of access rights in a delegation network. But before we can state the exact relation between delegation trees and the *authorizes* relation, we need the following lemma.

Lemma 16 *Let $\langle Nodes, Arcs, h \rangle$ be a delegation tree in a delegation network DN. A node is the root of an even-depth subtree iff h maps it into a key. Also, a node is the root of an odd-depth subtree iff h maps it into a certificate.*

Proof This is easily shown by induction on the length of subtrees. Basis step: all roots of subtrees of depth 0, i.e. the leaf nodes map into keys according to Lemma 14. The roots of subtrees of depth 1 are connected to the leafs and, by Lemma 13, mapped into certificates.

Induction step: assume that all roots of even-depth subtrees map into keys and roots of odd-depth subtrees map into certificates for subtrees of depth i

or smaller where i is even. Let n be a root of a subtree of depth $i+1$. n has a child that is a root of subtree of depth i and, by our assumption, maps into a key. By Lemma 13, n has to map into a certificate. On the other hand, let n be a root of a subtree of depth $i+2$. In that case, n has a child that is a root of a subtree of depth $i+1$ and, as concluded above, maps into a certificate. Again by Lemma 13, n has to map into a key. This shows that the roots of subtrees of depth $i+1$ map into certificates and of depth $i+2$ into keys. By induction, the assumption is valid for subtrees of any finite depth. Since all nodes of the tree are roots of subtrees of either even or odd depth, this suffices to prove the lemma. □

Finally, we are ready to show that the authorization problem can be formulated as a question on the existence of delegation trees. This is proven using Theorem 10 that said it is sufficient to look at finite subnetworks. The theorem is thus a consequence of the limitation for certificates to have only a finite number subjects.

Theorem 17 *Let $DN = \langle Keys, Certs, Auths, Flow, auth\rangle$ be a delegation network, o an operation in DN, and $k_1, k_2 \in Keys$. $authorizes_{DN}(k_1, k_2, o)$ is true iff there exists a delegation tree $DT = \langle Nodes, Arcs, h\rangle$ in DN such that*

1. *for the unique root node n of the tree, $h(n) = k_1$,*
2. *for all leaf nodes n of the tree, $h(n) = k_2$, and*
3. *for all nodes $n \in Nodes$, if $h(n) \in Certs$ then $o \in auth(h(n))$.*

Proof We first show that the existence of a delegation tree implies the authorization.

Let $DN = \langle Keys, Certs, Auths, Flow, auth\rangle$ be a delegation network and $DT = \langle Nodes, Arcs, h\rangle$ a delegation tree in DN such that the Conditions 1–3 of the theorem are satisfied. Every node of the delegation tree is the root of a subtree. We will show by induction on the depth of subtrees that $authorizes_{DN}(h(n), k_2, o)$ holds for all the nodes n that are mapped into keys by h. Basis step: Let n be a leaf node of DT. In that case, $h(n) = k_2$ and $authorizes_{DN}(k_2, k_2, o)$ by Condition 1 of Def. 4. Thus, the claim is true for all nodes that are roots of subtrees of depth 0.

Induction step: Assume that $authorizes_{DN}(h(n), k_2, o)$ for all nodes n that are roots of subtrees of even depth smaller than or equal to some even $i \geq 0$. Let n be the root of a subtree of depth $i+2$. Lemma 16 shows

that n is mapped into a key. Let n' be a child node of n. A child n' exists because $i + 2 > 0$. According to Lemma 13, the child is mapped into a certificate $h(n')$, and its children into keys. By Lemma 16, the children of n' are roots of subtrees of even depth. This depth is i or smaller. Therefore, the induction hypothesis implies that for all the children n'' of n', $authorizes_{DN}(h(n''), k_2, o)$. By Condition 3 of Def. 12, there exist children of n' mapped by h onto all of the subjects of $h(n')$. This means that $authorizes_{DN}(k, k_2, o)$ for all the subjects k of $h(n')$. Consequently, by Condition 2 of Def. 4, $authorizes_{DN}(k, k_2, o)$ where k is the issuer of $h(n')$. But by Condition 3 of Def. 12, the issuer is $k = h(n)$. That is, $authorizes_{DN}(h(n), k_2, o)$ for the root n of an arbitrary subtree of depth $i + 2$. By induction, $authorizes_{DN}(h(n), k_2, o)$ is true for all nodes in $n \in Nodes$ that map into keys, also for the root node that maps into k_1. Hence $authorizes_{DN}(k_1, k_2, o)$. This suffices to prove the 'if' direction of the theorem.

Next, we show that the authorization implies the existence of a delegation tree.

Let $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ be a delegation network and $authorizes(k_1, k_2, o)$ true. In Theorem 10 it was shown that DN has a finite acyclic subnetwork $DN' = \langle Keys', Certs', Auths, Flow', auth' \rangle$ where $authorizes(k_1, k_2, o)$ for all $k \in Keys'$. In the finite acyclic graph formed by the $Flow'$ relation there are no infinite chains. We will construct the finite delegation tree from this relation from root down.

Let $Nodes_0 = \{n_{-,k_1}\}$ and let $Arcs_0 = \emptyset$. Assign function h the value $h(n_{\emptyset,k_1}) = k_1$. For $i = 1, 2, \ldots$, let $Nodes_i = \{n'_{n,c}, n'_{n,k} \mid n \in Nodes_{i-1} \wedge \langle h(n), c \rangle \in Flow \wedge \langle c, k \rangle \in Flow\}$ where the nodes $n'_{n,c}$ and $n'_{n,k}$ are new nodes not in $Nodes_{i-1}$. Since the new nodes are named after their parent, the paths cannot join, and a tree is formed. Let also $Arcs_i = \{\langle n, n'_{n,c} \rangle, \langle n'_{n,c}, n'_{n,k} \rangle | n \in Nodes_{i-1} \wedge \langle h(n), c \rangle \in Flow \wedge \langle c, k \rangle \in Flow\}$. The construction follows certificate chains in DN' adding one key–certificate step on each iteration. Since the number of keys and certificates in DN' is finite and no loops exists, the construction must come to an end at some iteration after which $Arcs_i$ and $Keys_i$ are empty. Let j be the index of the last round where keys are found. There is only a finite number of nodes in all $Nodes_i$ because $Nodes_0$ is finite and, on every iteration, the number of nodes attached to each one of the previous nodes is limited by the finite number of keys and certificates in the network.

Let $Nodes = \cup_{i=0}^{j} Nodes_i$ and $Arcs = \cup_{i=0}^{j} Arcs_i$. These sets are also finite. Assign h the values $h(n'_{n,c}) = c$ and $h(n'_{n,k}) = k$ for all $n'_{n,c}, n'_{n,k} \in Nodes$. From the way the nodes were added to the sets, it follows that $\langle Nodes, Arcs \rangle$ is a tree, and h a homomorfism from the tree to DN'. This is because the

nodes mapping into certificates have one parent mapping into their issuer and a set of children corresponding to the subjects of the certificate. Thus, $DT = \langle Nodes, Arcs, h \rangle$ is a delegation tree in DN.

The root of the tree is n_{-,k_1} that is mapped into k_1 by h. Hence, Claim 1 of the theorem holds for the tree DT. According to Theorem 10 the subnetwork DN' can be selected in such a way that the only key that is not an issuer of any certificate in DN' is k_2, and that all certificates in DN' delegate the operation o. The former means that all leaf nodes of the constructed delegation tree DT map into k_2. The reason is that our construction of the delegation tree only ends at nodes that map into a key and whose corresponding key does not issue any certificates in DN'. (Def. 1 requires all certificates to have at least one subject). Thus, Claim 2 of the theorem holds for the tree DT. Since all nodes of the tree DT map into some key or certificate in DN', the latter means that the nodes can only map into certificates that delegate the right to operation o. Thus, also Claim 3 of the theorem holds. □

The trees are finite because we restricted the number of subjects on a certificate to finite. The same theorem would hold for infinite sets of subjects and infinite trees. The finiteness in the definition of delegation tree (Def. 15) could be replaced by a requirement that all paths from the root of the tree to the leafs have finite length. In real systems, however, finite sets of keys are more common, and we use the finite trees as a basis for terminating algorithms.

4 Certificate reduction

The SPKI draft document [15] presents a certificate reduction technique for authorization decisions. (It is called *5-tuple reduction* because the SPKI certificates are defined as 5-tuples). At the time being, the reduction is defined only for certificates with a single subject but we present our own definition that we believe to convey the idea accurately also for joint-delegation certificates. In fact, our definition is simpler because we do not need to distinguish between certificates with one and more subjects.

Sec. 4.1 contains the definition and illustration of the reduction technique. Sec. 4.2 shows rigorously that certificate reduction is a correct and adequate technique for making authorization decisions in our general framework.

4.1 Definition of certificate reduction

In certificate reduction, two certificates are merged into one. Fig. 4 illustrates the reduction process. The reduced certificate has the same issuer as the first of the original certificates and the combined subjects from both certificates, except for the one key that issued the lower certificate. This way, two certificates in a chain can be reduced into one. By repeating the process, any set of certificates can be combined into one.

Definition 18 (certificate reduction) *Let* $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ *be a delegation network. Delegation network* $DN' = \langle Keys, Certs', Auths, Flow', auth' \rangle$ *is obtained from* DN *by* reducing certificates c_1 *with* c_2, *iff* $Certs' = Certs \cup \{c\}$ *where* c *is a new certificate not in* $Certs$ *and*

$$\begin{aligned} Flow' = {} & \{\langle k, c\rangle \mid \langle k, c_1\rangle \in Flow\} \cup \\ & \{\langle c, k\rangle \mid \langle c_2, k\rangle \in Flow\} \cup \\ & \{\langle c, k\rangle \mid \langle c_1, k\rangle \in Flow \wedge \langle k, c_2\rangle \notin Flow\}. \end{aligned}$$

and $auth'(c) = auth(c_1) \cap auth(c_2)$.

It is important to note that reduction of c_1 with c_2 differs from the reduction of c_2 with c_1. When the names of the reduced certificates need not be explicitly mentioned, we simply say that DN' is obtained by a single certificate reduction from DN.

The definition allows the reduction of any two certificates, even when they do not form a chain. In practice, however, reductions are useful only when the issuer of c_2 is a subject of c_1.

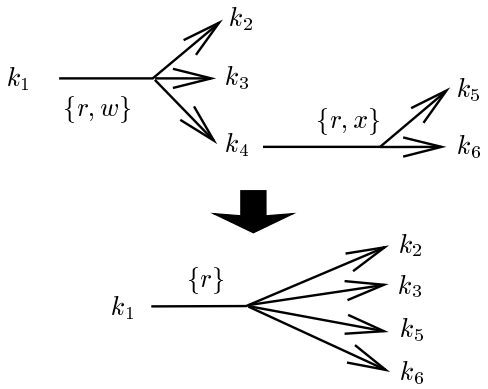

Figure 4: Certificate reduction

4.2 Soundness and completeness of certificate reduction

The next lemma shows that extending the subject set of a certificate does not increase the rights delegated to any key. That is, if two certificates are identical except that one has more subjects, that one is redundant.

Lemma 19 *Let $DN = \langle Keys, Certs, Arcs, Auths, auth\rangle$ be a delegation network and $DN' = \langle Keys, Certs', Arcs', Auths, auth'\rangle$ be a supernetwork of DN where $Certs' = Certs \cup \{c'\}$. Assume that the issuer of some $c \in Certs$ is also the issuer of c', $auth(c) = auth(c')$ and the set of subjects of c' is a superset of the set of subjects of c. Then, $authorizes_{DN'} = authorizes_{DN}$.*

Proof Since DN is a subnetwork of the delegation network DN', we have $authorizes_{DN} \subseteq authorizes_{DN'}$. For the inclusion in the other direction, assume that $authorizes_{DN'}(k_1, k_2, o)$. Theorem 17 says that there exist a delegation tree in DN' satisfying the three claims of the theorem. If there are nodes n in the tree that map into c', we redefine for all these nodes $h(n) = c$. Since the subjects of c' are a superset of the subjects of c, n has children mapping onto all the subjects of c. The children of n that do not map into subjects of c and the subtrees under these children are removed from the tree. By this construction, we get a delegation tree in DN for which the three claims of Theorem 17 still hold. Therefore, $authorizes_{DN}(k_1, k_2, o)$. This suffices to show the inclusion in the other direction. Thus, the relations are equal. □

Soundness of certificate reduction means that the reduced certificates do not have any effect on the *authorizes* relation in the delegation network. Completeness means that the reduction can be used as a way of deciding

the authorization problem. That is, it is possible to reduce any chain of delegation into a single certificate. The next lemma is essential in proving the soundness.

Lemma 20 *Let DN' be a delegation network obtained by a single certificate reduction from DN. Then, $authorizes_{DN'} \subseteq authorizes_{DN}$.*

Proof Let the delegation network DN' be obtained from DN by reducing certificate c_1 with c_2 whereby a reduced certificate c' is obtained. Assume that $authorizes_{DN'}(k_1, k_2, o)$. Theorem 17 says that there exist a delegation tree $DT' = \langle Nodes', Arcs', h' \rangle$ in DN' satisfying the three claims of the theorem.

There are three possible cases: (1) no nodes of DT map into c', (2) one or more nodes map into c' and the issuer of c_2 is a subject of c_1, and (3) one or more nodes map into c' and the issuer of c_2 is not a subject of c_1.

Case (1): If none of the nodes of the tree maps into c', the tree is also a delegation tree for DN and by Theorem 17, $authorizes_{DN}(k_1, k_2, o)$.

Case (2): The tree contains one or more nodes that map into the reduced certificate c'. Let n' be one of the nodes. In that case there exist also nodes n'_{iss} and $n'_{sub,i}$, $i = 1, \ldots, l$, mapping onto the issuer and the subjects of c'. We assume also that the issuer k of c_2 is a subject of c_1. We construct a new delegation tree by removing n' and adding two new certificate nodes n_1, n_2 and one key node n_3. $DT = \langle Nodes, Arcs, h \rangle$ and $Nodes = (Nodes' \setminus \{n'\}) \cup \{n_1, n_2, n_3\}$. The value of h is equal to h' for all nodes from DT', and for the new nodes, $h(n_1) = c_1$, $h(n_2) = c_2$ and $h(n_3) = k$, where k is the issuer of c_2. Denote the set of keys of DN by $Keys$. The new set of arcs is

$$\begin{aligned} Arcs = {} & ((Arcs' \setminus (Keys \times n')) \setminus (n' \times Keys) \cup \\ & \{\langle n'_{iss}, n_1 \rangle \langle n_1, n_3 \rangle, \langle n_3, n_2 \rangle\} \cup \\ & \{\langle n_1, n'_{sub,i} \rangle \mid i \in \{1 \ldots l\} \wedge \langle c_1, h(n'_{sub,i}) \rangle \in Flow\} \cup \\ & \{\langle n_2, n'_{sub,i} \rangle \mid i \in \{1 \ldots l\} \wedge \langle c_2, h(n'_{sub,i}) \rangle \in Flow\}. \end{aligned}$$

This construction gives a delegation tree in DN that still fulfills the three claims of Theorem 17. (The root and the leafs of the tree remain unchanged.) Consequently, $authorizes_{DN}(k_1, k_2, o)$.

Case (3): We still have to consider the possibility that the issuer of c_2 is not a subject of c_1. In that case, the new certificate c' is like c_1 only with extended set of subjects. By Lemma 19, this does not add anything new to the $authorizes_{DN'}$ relation.

Hence, the theorem holds in all cases. □

We now have the necessary tools for proving the main result of this section.

Theorem 21 (soundness and completeness of certificate reduction) *Let $DN_0 = \langle Keys, Certs, Auths, Flow, auth \rangle$ be a delegation network. It is true that $authorizes_{DN}(k_1, k_2, o)$ iff there is a finite sequence of delegation networks $DN_0, DN_1, DN_2, \ldots, DN_l$ such that DN_{i+1} is obtained from DN_i by certificate reduction for $i = 1, \ldots, l$, and that there is a certificate c in DN_l such that $o \in auth'(c)$ and the issuer of c is k_1 and the only subject of c is k_2.*

Proof

If DN_l has the certificate c described in the theorem, then by applying conditions 1 and 2 of Def. 4, we get $authorizes_{DN_l}(k_1, k_2, o)$. This must be true also in the original network DN_0 because from Lemma 20 it follows that $authorizes_{DN_{i+1}} \subseteq authorizes_{DN_i}$ for $i = 1, \ldots, l$ and consequently $authorizes_{DN_l} \subseteq authorizes_{DN_0}$. Hence, the reduction method gives sound results.

Let DN be a delegation network where $authorizes_{DN}(k_1, k_2, o)$. We need to show that there always is a finite sequence of certificate reductions that produce the certificate c.

Theorem 17 says that there exists a finite delegation tree DT_0 in DN_0 such that the three claims of the same theorem are satisfied. We claim that either (1) there is a node mapped into a certificate in DT such that its parent is mapped into k_1 and its only subject into k_2 and the certificate authorizes the operation o, or (2) there are two nodes n_1 and n_2 in DT_0 such that the parent of n_1 is a child of n_2. Assume that alternative (1) is not true. Then, select one leaf node n_{leaf} of the tree, and the parent n of this node. n maps into a certificate. If n has a child n' other than n_{leaf}, this child is not mapped into k_2 and thus is not a leaf and has a child n'' itself. In that case, we can choose $n_1 = n$ and $n_2 = n''$. On the other hand, if n has only n_{leaf} as a child, its parent node cannot be the root, because that would be case (1). Thus, the parent n' of n has a parent n'' and we can choose $n_1 = n''$ and $n_2 = n$. This shows that one of the alternatives (1) and (2) holds.

We now reduce pairs of certificates step by step and replace corresponding two nodes in the tree by new nodes corresponding to the reduced certificate. This way, we get a tree that shrinks in every reduction.

We start from DN_0 and its delegation tree DT_0 and for $i = 0, 1, 2, \ldots$, do the following. If alternative (1) does not hold but is true (2) instead, we can reduce the two certificates $h(n_1)$ and $h(n_2)$ where the issuer of the latter is a subject of the former. The reduction results in a new delegation network

DN_{i+1} with an added certificate c_{new}. We construct a delegation tree DT_{i+1} by removing the nodes n_1, n_2 and the node n_3 corresponding to the issuer of $h(n_2)$ from the tree and by inserting a new node n_{new} instead. n_{new} has the issuer of n_1, and all the children of n_1 and n_2 except for n_3. We also assign $h(n_{new}) = c_{new}$. The resulting tree DT_{i+1} is a delegation tree in DN_{i+1}, because the new node corresponds to the reduced certificate. Furthermore, DT_{i+1} also fulfills the three claims of Theorem 17 since the root and leafs do not change and $o \in auth(c_{new}) = auth(h(n_1)) \cap auth(h(n_2))$.

This way we get a sequence of delegation networks $DN_0, DN_1, DN_2, \ldots$ and trees in them $DT_0, DT_1, DT_2, \ldots$. Since DT_0 is finite and in every reduction two nodes of the previous tree are replaced with one in the next tree, the process has to end at some point in alternative (2) becoming false. This happens at latest when there is only once certificate node left in the tree. Hence, for some $l \geq 0$ the alternative (1) will be true and the desired c exists in DN_0.

When alternative (1) holds in the tree DN_l, we have the desired result. That is, there is a single certificate c in DT_l as described in the theorem above. $\square$

5 Threshold certificates

In this section we describe certificates where a sufficiently large subset of the subjects of a certificate can delegate or use the authority given by it. Sec. 5.1 introduces threshold values and Sec. 5.2 describes how threshold certificates can be made more flexible by dividing them into subcertificates.

5.1 (k, n) schemes

A (k, n)-threshold certificate is considered valid if k of its n subjects co-operate in using or further delegating the access rights. Joint-delegation certificates with k subjects correspond to (k, k)-threshold certificates. The threshold value is simply a convenient short-hand notation for a set of joint-delegations where all subjects are required to co-operate. That is, a (k, n)-threshold certificate can be expanded to $\binom{n}{k}$ joint-delegation certificates with k subjects in each. Therefore, we have not complicated the theory above with threshold values.

5.2 Open threshold certificates

In the joint-delegation and threshold certificates described above, the set of subject keys has to be fixed at the time of certificate creation. This is because the keys are explicitly listed in the certificate. It is, however, possible to leave the set of subjects open. We can give each subject a separate certificate (*subcertificate*) that includes the threshold value and a unique identifier of the certificate set. The set of certificates are considered valid only if the threshold number of subcertificates with the same identifier are presented together. This way, the set of subjects is open for later additions. Moreover, the division of the certificates into several subcertificates adds flexibility to certificate management and the holders of the certificates can remain anonymous until they want to further delegate their share of the access rights. We call this kind of scheme *open threshold certificates*. The properties of the open threshold certificates make them an attractive alternative to fixed subjects sets. This is especially so because it appears that most implementations would be simplified by the transition.

In this section, we will show that the open threshold certificates can simulate the functionality of normal threshold and joint-delegation certificates and that the security of the system is not endangered in the transition.

First, open threshold certificates must be formally defined. We do this by adding "dummy" operations to the delegation network and by redefining the

authorizes relation.

Definition 22 (open-threshold-type authorizations) Open-threshold-type authorizations *are triples* $\langle id, l, a\rangle \in Ids \times \mathbb{Z}^+ \times Auths$ *where Ids is a set of identifiers,* $\mathbb{Z}^+$ *are positive integers called* threshold values *and Auths are set-type authorizations.*

The authorizations of the form $\langle id, 1, a\rangle$ *with any value of id are identified with each other for every a and the symbol a is used to represent them.*

The set of *operations* for a delegation network with open-threshold-type authorizations *OAuths* is defined as $Ops = \cup\{a \mid \langle id, l, a\rangle \in OAuths\}$. The new fields *id* and l in the certificates are used to convey information about joint delegation and the field a gives the set operations for which rights are being delegated.

We need to define the *authorizes* relation for the new type of authorizations.

Definition 23 (*authorizes* relation) *Let* $DN = \langle Keys, Certs, OAuths, Flow, auth\rangle$ *be a delegation network where the authorizations are of open-threshold type. Denote by Ops the operations of DN. The authorization relation* $authorizes_{DN} \subseteq Keys \times Keys \times Ops$ *is the smallest three-place relation such that*

1. *if* $k \in Keys$ *and* $o \in Ops$*, then* $\langle k, k, o\rangle \in authorizes_{DN}$*, and*
2. *if for some* $id \in Ids$*,* $l \in \mathbb{Z}^+$ *and* $k_1 \in Keys$ *there exist at least l pairs of keys k and certificates c such that* $\langle k, k_2, o\rangle \in authorizes_{DN}$*,* $\langle c, k\rangle \in Flow$*,* $\langle k_1, c\rangle \in Flow$ *and* $auth(c) = \langle id, l, a\rangle$ *where* $o \in a$*, then* $\langle k_1, k_2, o\rangle \in authorizes_{DN}$*.*

In practice, there should be a single threshold value matching each identifier and only one key should issue certificates with a given identifier. Since these rules cannot be enforced in a distributed system of issuers, the definition above treats certificates with equal identifier but differing threshold value or issuer as belonging to different groups, just as if they had different identifiers. Other policies can of course be defined for this kind of situations.

Finally, we are able to give a transformation from delegation networks with joint-delegation or threshold certificates to ones with open threshold certificates. The resulting network simulates a joint-delegation certificate by issuing to all subjects separate certificates that contain a common identifier.

Definition 24 (transformation *Open*) *Let* $DN = \langle Keys, Certs, Auths, Flow, auth\rangle$ *be a delegation network with set-type certificates.* $Open(DN) =$

$\langle Keys, Certs', OAuths, Flow', auth' \rangle$ is a delegation network defined by

$$\begin{aligned} Certs' &= \{c'_{c,k} \mid c \in Certs \wedge \langle c, k \rangle \in Flow\}, \\ Flow' &= \{\langle k', c'_{c,k} \rangle \mid c'_{c,k} \in Certs' \wedge \langle k', c \rangle \in Flow\} \cup \\ &\quad \{\langle c'_{c,k}, k \rangle \mid c'_{c,k} \in Certs'\}, \\ OAuths &= Certs \times \mathbb{Z}^+ \times Auths. \end{aligned}$$

For all $c'_{c,k} \in Certs'$, $auth'(c'_{c,k}) = \langle c, l, auth(c) \rangle$ where l is the number of subjects of the certificate c.

It should be carefully noted that the certificates $c'_{c,k}$ are just plain items in the certificate set. In implementations, they will not contain any identification of the original c and k. The new authorizations, on the other hand, have an explicit field containing the name of the original certificate or other unique identifier for the set of certificates in $Open(DN)$ that are derived from one certificate in DN. Since the original certificate names do not have any structure and they are forgotten in the transformation process, this field does not carry any hidden knowledge of the structure of the original network. It only groups the new certificates according to their origin.

We will show that the transformation *Open* preserves the *authorizes* relation. This means that the open threshold certificates can express any kind of delegation that the set-type authorizations could.

Theorem 25 *Let DN be a delegation network with set-type authorizations. Then,*

$$authorizes_{DN} = authorizes_{Open(DN)}.$$

Proof Let $DN = \langle Keys, Certs, Auths, Flow, auth \rangle$ be a delegation network with set-type authorizations and let $Open(DN) = \langle Keys, Certs', Flow', OAuths, auth' \rangle$. We notice that a certificate in DN corresponds to a set of certificates in $Open(DN)$. This set is the certificates that were named $c'_{c,k}$ for the subjects k of c.

If we consider the *authorizes* relations in the two networks, we see that Defs. 4 and 23 both define *authorizes* as a closure of the set defined by Rule 1, on which the two definitions agree, with respect to Rule 2, which differs in the definitions. We will compare Rules 2 and see that they actually are equivalent. Assume that $authorizes_{DN}(k, k_2, o)$ and $authorizes_{Open(DN)}(k, k_2, o)$ for all keys k in some set K.

Assume also that $\langle k_1, c \rangle \in Flow$, that $o \in auth(c)$ and that all the keys k for which $\langle c, k \rangle \in Flow$ are in K. The idea of the assumption is that the

conditions of Rule 2 of Def. 4 are met. By Def. 24, the number l of certificates corresponding to c in $Open(DN)$ is equal to the number of subjects of c. This l is also the threshold number visible in the authorizations of all the l certificates. The l certificates have all k_1 as issuer and the subjects of c as subjects. Since all these subjects are in K, the conditions listed in Rule 2 of Def. 23 are also met.

On the other hand, assume that in $Open(DN)$ for some $id \in Ids$ and $l \in \mathbb{Z}^+$ there exist at least l pairs of certificates c and keys $k \in K$ such that $\langle k_1, c\rangle \in Flow'$, $\langle c, k\rangle \in Flow'$ and $auth'(c) = \langle id, l, a\rangle$ where $o \in a$. This assumption has the meaning that the conditions of Rule 2 of Def. 23 are met. Again, Def. 24 requires that there is at least one certificate in DN with the same issuer k_1 and the same l subjects. The reason is that the values of id in $Open(DN)$ uniquely identify a group of certificates corresponding to one certificate in DN. Since all the subjects k are in K, the conditions listed in Rule 2 of Def. 4 are fulfilled.

Hence, Rule 2 in one of the definitions is applicable to a key k_1 if and only if it is applicable in the other definition. As the closure rules are equal and the starting sets are equal, the resulting closures are also equal. □

We will denote the issuers of a set of certificates by $issuers(C) = \{k \mid c \in C \wedge \langle k, c\rangle \in Flow\}$.

Next we want to show that addition or removal of certificates in DN can be simulated by addition or removal of certificates by the same issuers in $Open(DN)$. This proves that the transformation $Open$ preserves the functionality of the delegation network.

Theorem 26 *Let DN_1 be a delegation network with certificates $Certs_1$ and with set-type authorizations and let DN_2 with certificates $Certs_1$ be its subnetwork. Denote the certificates of $Open(DN_1)$ by $Certs'_1$ and of $Open(DN_2)$ by $Certs'_2$. Then, $Open(DN_2)$ is a subnetwork of $Open(DN_1)$, and*

$$issuers(Certs_1 \setminus Certs_2) = issuers(Certs'_1 \setminus Certs'_2).$$

Proof That $Open(DN_2)$ is a subnetwork of $Open(DN_1)$ is a direct consequence of the monotonic nature of the transformation $Open$. Added certificates in DN result in added certificates in $Open(DN)$. The issuers of the added certificates are also the same. □

In order to see that the transformation is secure, we still must show that any additions to the *authorizes* relation that can be achieved by a set of

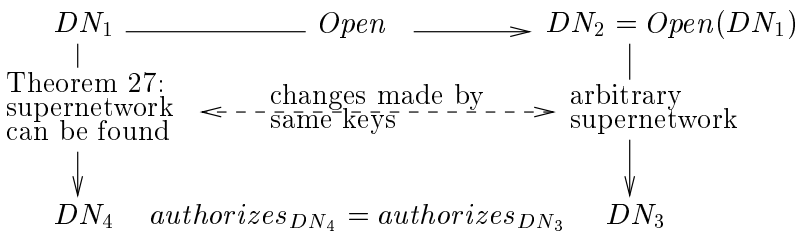

Figure 5: Delegation networks in Theorem 27

keys in $Open(DN)$ could also be caused by the same set of keys in DN. When issuing new access rights in $Open(DN)$ can be simulated in DN by the same issuers, we know that the transformation does not endanger the access control policy.

Theorem 27 *Let DN_1 be a delegation network with set-type authorizations and certificates $Certs_1$, and denote the certificates of $DN_2 = Open(DN_1)$ by $Certs_2$. Let DN_3 be a supernetwork of DN_2 with the same set of keys and authorizations. Then, DN_1 has a supernetwork DN_4 with certificates $Certs_4$ such that*

$$authorizes_{DN_3} = authorizes_{DN_4} \quad \textit{and}$$

$$issuers(Certs_4 \setminus Certs_1) = issuers(Certs_3 \setminus Certs_2).$$

Proof We first include in DN_4 all the certificates of DN_1. Let then $C = Certs_3 \setminus Certs_2$ be the set of added certificates in DN_3. If $auth_3(c) = \langle id, l, a \rangle$ for a certificate $c \in C$, then we add to DN_4 a certificate for every set of l certificates in DN_3 whose identifier is id, threshold value l and issuer the same as that of c. The subject sets of these certificates are formed by the subjects of the l certificates. Clearly, the issuers of the certificates $Certs_4 \setminus Certs_1$ will be the same keys as the issuers of the certificates C.

If we now compute $Open(DN_4)$, the result is almost equal to DN_3. One difference is that the identifiers of certificate groups may have changed and that some groups may have been duplicated. Another difference is that if new certificates were added with an identifier already existing in DN_3 thus exceeding the threshold value l associated with the identifier (this is a (l, n) scheme with $n > l$), the subsets of size l of the certificates have been enumerated as certificate groups of size l with new identifiers. The changes of identifiers and duplication of certificates naturally does not affect the *authorizes* relation. Also, the splitting of certificate groups to all their threshold-size subsets does not cause any changes to the situations where Rule 2 of Def. 23 can be applied.

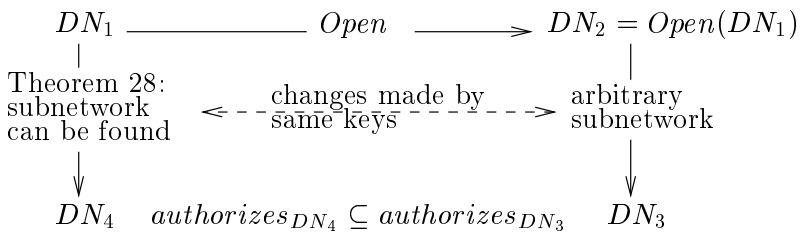

Figure 6: Delegation networks in Theorem 28

As in the proof of Theorem 25, closure of the same base set with respect to the same rule results in the same *authorizes* relation in DN_3 and $Open(DN_4)$. Hence, $authorizes_{DN_3} = authorizes_{Open(DN_4)} = authorizes_{DN_4}$. □

Similarly, it is possible to show that removal of certificates from $Open(DN)$ can be simulated or surpassed in DN by removal of certificates issued by the same key. This means that the transformation does not open any new lines of denial of service attack by expiring or revoking certificates.

Theorem 28 *Let DN_1 be a delegation network with set-type authorizations and certificates $Certs_1$, and denote the certificates of $DN_2 = Open(DN_1)$ by $Certs_2$. Let DN_3 be a subnetwork of DN_2 with the same set of keys and authorizations. Then, DN_1 has a subnetwork DN_4 with certificates $Certs_4$ such that*

$$authorizes_{DN_4} \subseteq authorizes_{DN_3} \quad \textit{and}$$

$$issuers(Certs_1 \setminus Certs_4) = issuers(Certs_2 \setminus Certs_3)\ )$$

Proof Denote be the set of removed certificates in DN_3 by $C = Certs_3 \setminus Certs_2$. We construct DN_4 by removing from DN_1 all certificates issued by the keys $issuers(C)$. All certificates issued by other keys are retained. Clearly, the issuers of the certificates $Certs_4 \setminus Certs_1$ are the same keys as the issuers of the certificates C.

In a way similar to the proof of Theorem 25, we show that if Rule 2 of Def. 4 is applicable in DN_4 then Rule 2 of Def. 23 is applicable in DN_3. Write $DN_3 = \langle Keys, Certs, Auths, Flow, auth\rangle$ and $DN_4 = \langle Keys, Certs', Auths', Flow', auth'\rangle$.

Let K be a set of keys where for all keys k both $authorizes_{DN_3}(k, k_2, o)$ and $authorizes_{DN_4}(k, k_2, o)$. Assume that for some certificate c, $\langle k_1, c\rangle \in Flow'$, $o \in auth'(c)$ and $\langle c, k\rangle \in Flow'$ implies $k \in K$. The assumption means

that $authorizes_{DN_3}(k_1, k_2, o)$ by Rule 2 of Def. 4. Then c is also in DN_1 and not issued by any key in $issuers(C)$. If c has l subjects, there are l certificates in DN_2 corresponding to c. These l certificates have a common identifier, they are all issued by the issuer of c, their threshold value is l, and they have the subjects of c as their subjects. Since the issuer of c is not in C, these certificates remain in DN_3. Hence, Rule 2 of Def. 23 says that $authorizes_{DN_3}(k_1, k_2, o)$.

$authorizes_{DN_3}$ and $authorizes_{DN_4}$ are closures of the same set ($\{\langle k, k\rangle \mid k \in Keys\}$ of Rule 1) with respect to the different Rules 2. Since Rule 2 is applicable in DN_3 always when it is in DN_4, the closure in DN_4 is a subset of the closure in DN_3: $authorizes_{DN_4} \subseteq authorizes_{DN_3}$. □

It should be noted that Theorems 25–27 and the proofs of this section do not only show properties of our proposed certificates scheme. They can generally be used as guidelines as to what kind of properties must always be shown when we want to replace a certificate scheme by another without changing the security properties.

6 Algorithms for deciding the authorization problem

In the literature, no actual algorithms for authorization decisions have been described. The SPKI document [15] states that its authors believe that the authorization problem can be answered but no implementation exists for the time being. Some discussion on the implementation techniques but no complete algorithms can be found in [22, 19].

In this section, we will describe several algorithms for the authorization decisions. The algorithms are designed to handle threshold certificates. Normal joint-delegation certificates are special cases where the threshold value equals the number of subjects.

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 quadratic with respect to the size of the certificate database. Using that much storage space does not seem feasible in the implementations that we have in mind, although some kind of caching might improve the efficiency of our algorithms. Secondly, signature verification is not a part of the algorithm. All signatures are verified at the time when the certificates are entered into the database.

6.1 Typical delegation network structure

The delegation networks in practice will not be arbitrary graphs but they are expected to have certain structure. Although the system architecture itself does not constrain the relations 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. 7). On the top of the hourglass there are the servers and on the bottom 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 certificate managers 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. 9(a).

Application programs, user platforms and servers themselves are unlikely to incorporate wide capabilities for maintaining valid certificates. There-

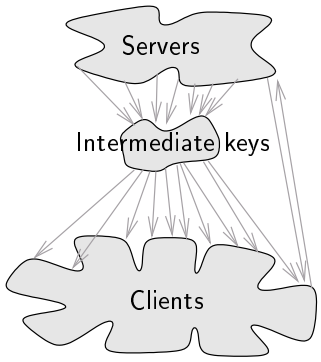

Figure 7: Typical delegation network structure

fore, specialized server software is needed for certificate acquisition, updates, bookkeeping and verification. These servers will need algorithms for authorization decisions 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 issued by clients to servers. Also, the servers or clients can create a wealth of relationships amongst themselves. Thus, the system must be able to accommodate certificates between arbitrary keys. Nevertheless, we will optimize the efficiency of our algorithms with the hourglass structure in mind.

6.2 Certificate reduction as an algorithm

The only decision procedure defined in the SPKI document is certificate (i.e. five-tuple) reduction. That is, rules are given for how two certificates reduce into one. A server should grant access to a client if there is a path of certificates that recursively reduces into a single certificate where the server itself authorizes the client. The reduction rules correspond to our definition of certificate reduction in Def. 18.

Since the reduction technique is a sound and complete (see Theorem 21) way of deciding the authorization problem, it can be used as an implementation technique. When a client has two certificates that form a chain, it sends them to the issuer of the first one who signs and returns a reduced certificate. For the issuer, the operation is purely syntactic manipulation of the certificates. It checks the signatures and automatically grants the reduced signature. The client has certificates reduced when possible or when it thinks it needs to. When the client wants to request an operation from a server, it sends along the request a reduced certificate signed by the server that directly authorizes

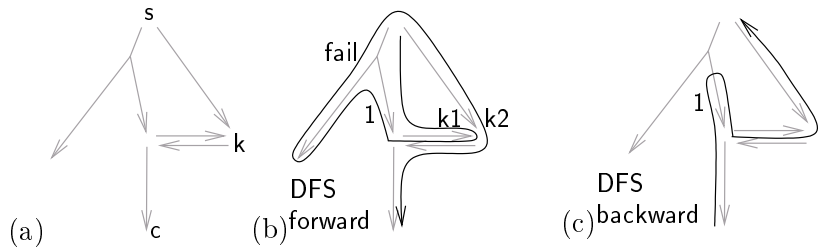

Figure 8: Depth-first forward search can visit the same node several times.

the client to the operation.

A problem with this approach is that the client must maintain the certificates and decide when it needs to start reducing the path. Therefore, techniques are needed for efficient path finding and decision making from a set of certificates even when the certificate reduction is actually implemented.

6.3 Depth-first search forward

The most straight-forward way to verify authorization from a delegation network is depth-first search in the graph of keys and certificates tracing the flow of access rights from the server to the client. The recursive search procedure has, in fact, been proposed as an alternative definition of authorization for the SPKI certificates [29].

Pseudo-code for a recursive depth-first search algorithm is found in Listing 1. The algorithm should contain no surprises to the reader. For each certificate, it counts the valid 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 and equals the number of subjects for normal 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. 8(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. 6.6, 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.

```
 1 function dfsForward (server, client, operation)
 2     return dfsForwardRecursive(server, client, operation);
 3
 4 function dfsForwardRecursive (key, client, operation)
 5    mark key as in search path;
 6    if (key = client) return TRUE;
 7    for c in certificates issued by key
 8       if (c authorizes operation)
 9          countPaths = 0;
10          for (subj in subjects of c)
11             if (countPaths < threshold value of c
12                AND (subj marked as having path to client
13                   OR (subj NOT marked as having path to client
14                      AND subj NOT marked as in search path
15                      AND dfsForwardRecursive(subj, client,
                                                operation))))
16                countPaths = countPaths + 1;
17          if (countPaths ≥ threshold value of c)
18             mark key as having path to client;
19             return TRUE;
20    unmark key as in search path;
21    return FALSE;
```

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

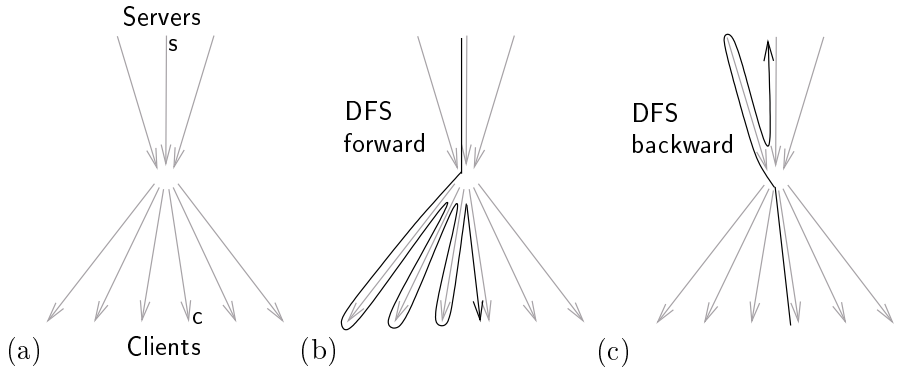

Figure 9: When forward branching is greater, backward search is faster

6.4 Depth-first and breadth-first search backward

In addition to the multiple traversing of same paths, the forward search has another inefficiency. In Fig. 9, 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.

There are two possible ways for handling joint-delegation certificates in backward search. 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. 8 (c) illustrates how the backward search processes every key at most once.

Thus, the backward search in a delegation network with many joint-delegation certificates is much more efficient than the forward search, linear-time with respect to the size of the network. Because of the different branching factors, it is also somewhat more efficient in a delegation network where all certificates have a single subject.

Backward search can also be done in breadth-first order as in Listing 3. The

```
22 function dfsBackward (server, client, operation)
23    return dfsBackwardRecursive(client, server, client, operation);
24
25 function dfsBackwardRecursive (key, server, client, operation)
26    mark key as having path to client;
27    if (key = server) return TRUE;
28    for c in certificates given to key
29       if (c authorizes operation
30          AND issuer of c NOT marked as having path to client)
31          countPaths = 0;
32          for (subj in subjects of c)
33             if (subj marked as having path to client)
34                countPaths = countPaths + 1;
35          if (countPaths ≥ threshold value of c
36             AND dfsBackwardRecursive(issuer of c, server,
37                                      client, operation)
38             return TRUE;
39    return FALSE;
```

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

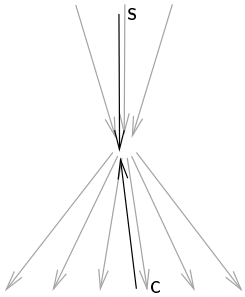

Figure 10: Two searches meet in the middle

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.

6.5 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. 10.

The average cost c of finding the path between two nodes in a graph grows exponentially with the length d of the path. 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 general 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 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

```
40 function bfsBackward (server, client, operation)
41    nextKeys = {client};
43    mark client as having path to client;
44    while (nextKeys != ∅)
45       currentKeys = nextKeys;
46       nextKeys = ∅;
47       for key in currentKeys
48          for c in certificates given to key
49             if (c authorizes operation
50                AND issuer of c NOT marked as having path
                                                 to client)
51                countPaths = 0;
52                for subj in subjects of c
53                   if (subj marked as having path to client)
54                      countPaths = countPaths + 1;
55                if (countPaths ≥ threshold value of c)
56                   if (number of certificates given to
                                        issuer of c > 0)
57                      nextKeys = nextKeys ∪ {issuer of c};
58                   mark issuer of c as having path to client;
59                   if (issuer of c = server) return TRUE;
60    return FALSE;
```

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

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

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

away from the end from which the branching is β_1.

In practice, the average paths are likely to be so short that the above formula is not needed to determine the optimal depth of the forward search. Instead, a constant value of one or two can be used unless the delegation paths are known to be especially long.

In a delegation network with a lot of joint-delegation certificates, the gains of two-way searching are not as big as suggested by the above formula. The reason is that the joint-delegation certificates make forward search of more than one or two steps from the server towards the client infeasible. Still, 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.

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 search from the client. The forward search is terminated at a specified maximum depth and at joint-delegation certificates. This way, it visits every key only once.

6.6 Experimental evaluation of the algorithms

Experiments conducted with generated certificate data confirm that the two-way search is the most efficient of the algorithms. The backward search algorithms appear almost as fast.

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 creating the certificates between random keys in the specified levels.

The data presented here was collected from a network with 4 layers of keys.

Level	# keys
1	100
2	10
3	100
4	5000

# certificates		from level			
		1	2	3	4
	1	5	200	10	100
to level	2	2	2	200	10
	3	2	2	5	20000
	4	2	2	2	500

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

Table 1: Parameters for the generated delegation network

	Search algorithm			
Decision	dfs forward	dfs backwd+forwd	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 for a key pair in different algorithms

Table 1 shows the number of keys on each level and the certificates between them. It should be noted that in this 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 had great effect on the efficiency of the forward depth-first search (see the discussion below). Although we have chosen the parameters to somewhat help that algorithm, the results are not too favorable to it.

The experiments with different one-way algorithms showed that the depth-first backward search and breadth-first backward search perform best (see Table 2). Any performance differences between these two algorithms were in-

	Depth of forward search				
Decision	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

	Depth of forward search				
Decision	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

significant 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 completeness of the graph and on the number of backward arcs 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 a 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 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.

The two-way search was tested by the first starting depth-first forward from the server to a maximum depth, and then looking for for the marked nodes

with breadth-first search backward from the client. (The depth-first search stopped at all joint-delegation certificates; see Sec. 6.5.) Table 3 shows how the cost of computation varied in the two-way search as a function of the depth of the forward search.

With the delegation network of four levels, one step of forward search gave best results. This is so because the one step away from the client saves a lot of work in going through the large number servers attacked to a single broker. Nevertheless, the average 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 two certificates. When the network had five or more levels of keys, forward search of depth two was usually faster. The savings in computation time were between 10 and 50 %.

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 when joint-delegation certificates are present. This is natural because the forward search part in the two-way search cannot handle 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.

7 Conclusion

In this report, we presented a formal model of distributed, key-oriented trust management systems. In the new systems, emphasis is put on delegation of access rights with certificates. In contrast to centralized or hierarchical, identity-oriented systems, the new distributed access control systems allow free formation of local and global authorities and trust relations between them. Rights are connected to cryptographic keys, not to identities of persons or systems. This approach allows much more freedom in limiting the level of trust between entities. As far as we know, our work is the first attempt to formalize the ideas behind these systems. It appears that the concept of delegation can be presented as a relatively simple formal model without consideration to implementation details.

In particular, we presented a formal semantics of delegation in a network of certificates. The delegation network was defined as a bipartite graph whose nodes are keys and certificates. The arcs of the graph represent the flow of authority from issuer keys to certificates and from certificates to subject keys. The main question to be queried from a delegation networks is that does a key authorize another one to a given operation with a given set of certificates. It was shown that when each certificate has only a finite number of subjects, the authorization of a key by another one is always done with a finite set of the certificates. We also gave an equivalent tree-based formulation of authorization. This made it easy develop intuitive proofs and to visualize the workings of algorithms.

The biggest advantage of the formal model was that it made it possible to discuss general properties of delegation networks without considering the details of various standards proposals. The equivalence of different techniques for access control decisions was proven. In particular, we proved the soundness and completeness of the SPKI certificate reduction with respect to the model. Moreover, we suggested a simple way for representing threshold certificates and proved it to have desired functional and security properties. Hopefully, the proposed changes will have effect on the ongoing standardization work.

The model was also used as a basis for development of algorithms for managing certificate databases. 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. Conceptual analysis and measurements on generated certificate data were done to compare the efficiency of the algorithms. The main observations from the experiments 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 from the client key towards the server.

In the future, we hope to derive a further abstraction of the formal model where the authorizations can have general lattice structure instead of being rights to perform a set of operations. This should make the theory simpler and mathematically more aesthetic. Other promising lines of future work include implementation of a certificate management database and development of algorithms for automatic retrieval of certificates from a network environment.

References

[1] Martín Abadi. On SDSI's linked local name spaces. In *Proc. 10th IEEE Computer Security Foundations Workshop*, pages 98–108, Rockport, MA, June 1997. IEEE Computer Society Press.

[2] Edward Amoroso. *Fundamentals of Computer Security Technogy*. Prentice Hall, 1994.

[3] Tuomas Aura. Comparison of graph-search algorithms for authorization verification in delegation networks. In *Proc. 2nd Nordic Workshop on Secure Computer Systems NORDSEC'97*, Espoo, Finland, November 1997.

[4] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations and model. Technical Report M74-244, The Mitre Corp., Bedford MA, May 1973.

[5] K. Biba. Integrity considerations for secure computer systems. Technical Report MTR-3153, The Mitre Corp., Bedford, MA, 1975.

[6] Matt Bishop. Theft of information in the take-grant protection model. *Journal of Computer Security*, 3(4), 1996.

[7] B. Blakley and D. M. Kienzle. Some weaknesses of the TCB model. In *Proc. 1997 IEEE Symposium on Security and Privacy*, pages 3–5, Oakland, CA, May 1997. IEEE Computer Society Press.

[8] 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.

[9] David F. Brewer and Michael J. Nash. The Chinese wall security policy. In *Proc. IEEE Symposium on Security and Privacy*, pages 206–214, Oakland, CA, May 1989. IEEE Computer Society Press.

[10] *Recommendation X.509, The Directory - Authentication Framework*, volume VIII of *CCITT Blue Book*, pages 48–81. CCITT, 1988.

[11] D. W. Chadwick, A. J. Young, and N. Kapidzic Cicovic. Merging and extending the PGP and PEM trust models — the ICE-TEL trust model. *IEEE Network*, pages 16–24, May 1997.

[12] William Cheswick and Steven Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, 1994.

[13] D. Clark and D. Wilson. A comparison of commercial and military computer security policies. In *Proc. IEEE Symposium on Security and Privacy*, Oakland, CA, 1987. IEEE Computer Society Press.

[14] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* MIT Press, 1990.

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

[16] Fred Halsall. *Data Communications, Computer Networks and Open Systems.* Addison-Wesley, 4th edition, 1996.

[17] Recommendation X.509 (11/93) - the directory: Authentication framework. ITU, November 1993.

[18] J. Kohl and C. Neuman. The Kerberos network authentication service (V5). RFC 1510, IETF Network Working Group, September 1993.

[19] Ilari Lehti and Pekka Nikander. Certifying trust. Unpublished manuscript, November 1997.

[20] John McLean. Is the trusted computing base concept fundamentally flawed? In *Proc. 1997 IEEE Symposium on Security and Privacy*, page 2, Oakland, CA, May 1997. IEEE Computer Society Press.

[21] Alfred J. Menezes, Paul C. van Oorschot, and Scot A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[22] Pekka Nikander. A Java Beans component architecture for cryptographic protocols. In *Proc.7TH USENIX Security Symposium*, San Antonio, Texas, January 1997. To appear.

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

[24] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons, Inc., 1996.

[25] William A. Shockley and James P Downey. Is the reference monitor concept fatally flawed? A case for the negative. In *Proc. 1997 IEEE Symposium on Security and Privacy*, pages 3–5, Oakland, CA, May 1997. IEEE Computer Society Press.

[26] William Stallings. *Network and Internetwork Security.* Prentice Hall, 1995.

[27] William Stallings. *Protect Your Privacy: A Guide for PGP Users.* Prentice Hall, 1995.

[28] Margaret Schlosser Wu. *Hierarchical Protection Systems.* PhD thesis, University of Iowa, December 1980.

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

[30] Andrew Young. Trust models in ICE-TEL. In *Proc. Internet Society Symposium on Network and Distributed System Security*, San Diego, CA, February 1997.

[31] Philip Zimmermann. *The Official PGP User's Guide.* MIT Press, June 1995.