On the Structure of Delegation Networks

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

Abstract

In new distributed, key-oriented access control systems such as SPKI, access right are delegated by a freely formed network of certificates. 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 co-operate to 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.

1 Introduction

New key-oriented access control systems offer a fully distributed alternative to traditional hierarchical, identity-oriented schemes. In the new systems, access right 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 of trust relations between the keys. This way, local authorities are free to form trust relations without the need for the kind of global hierarchy of trusted servers as, for example, in X.509 [4].

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 the delegation.

The model is used for proving the equivalence of several methods for access control decisions. In particular, we show that the certificate reduction technique of [5] 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 made it possible to develop efficient algorithms for access control decisions from a database of certificates in [2].

Notable differences between out model and traditional capability-based security models [9] are that the delegation certificates can be issued by anyone without special "grant" rights and even before the issuer itself has obtained the delegated rights. Instead, the delegated rights are limited to reflect the level of trust to the receiving keys. Moreover, the complexity of our model arises from joint-delegation and threshold certificates, i.e. certificates that are issued to a set of subjects who must co-operate to use the authority.

We represent the delegation networks as graphs and make access control decisions by finding paths in the graphs. An alternative approach could be to define a logical language for describing delegation and to make decisions by proving theorems of the logic. It appears that the access control calculus of Abadi et al. [1, 6, 10] could be adapted for the purpose. However, the calculus was developed before key-based systems and it does not directly support anonymous keys in the certificate chain.

The three most prominent proposals for distributed trust management on open communications networks are SPKI certificates [5] by Ellison et al., SDSI public key infrastructure [8] by Rivest and Lampson, and PolicyMaker local security policy database [3] by Blaze et al. In the development of our theory, we have most often referred to the SPKI specification.

We begin by defining delegation network and discussing subnetworks in Sec. 2. Sec. 3 shows how the delegation can be visualized as trees. The soundness and completeness of certificate reduction are proven in Sec. 4. In Sec. 5 we suggest a slightly generalized version of SPKI threshold certificates that adds flexibility to certificate management.

2 Delegation network

We start by defining a structure called *delegation network* in Sec. 2.1. It consists of keys and certificates for delegating authorizations between the keys. The authorization

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

are rights to perform sets of operations. This is detailed in Sec. 2.2. In Sec. 2.3 we continue by formulating the authorizations 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 are presented in Sec. 2.4.

2.1 Definition of delegation network

We define a delegation network as a directed bipartite graph. In a bipartite graph, the nodes are divided into two partitions and all arcs are drawn between the partitions. In a delegation network, the partitions 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. At our level of abstraction, the keys are primitive data items. 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 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.

When a certificate has more than one subject, the idea is that all of them must co-operate to use the authority given by the certificate. Normally they do this by further delegating the authority to a single key. Hence, a certificate is made weaker by adding subjects.

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 a computer network can be unlimited.

In practice, the certificates may have other fields such as limitations on the propagation of the access rights to other subjects. Adding a no-redelegation bit to the model would be straightforward but we have chosen to leave it out for simplicity. In any case, the primary mechanisms for parameterizing trust to the subject is to limit the value in the authorization field of the certificate.

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 annotated arcs between the keys.

Note also that we allow delegation networks to have cycles, i.e. a key can directly or indirectly delegate 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. To avoid complexity in definitions, we do not want to disallow even direct delegation to self although it is never useful in practice. We will, however, show (in Theorem 8) 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*
$\langle k_1, c_1 \rangle, \langle c_1, k_2 \rangle, \langle k_2, c_2 \rangle, \langle c_2, k_3 \rangle, \ldots, \langle c_{n-1}, k_n \rangle \in Flow$ *where* $k_1 = k_n$*.*

Fig. 1 shows an example of a delegation network. For simplicity, the certificate nodes are not explicitly drawn. The arrows in the figure represent both certificates and their incoming and outgoing arcs. 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

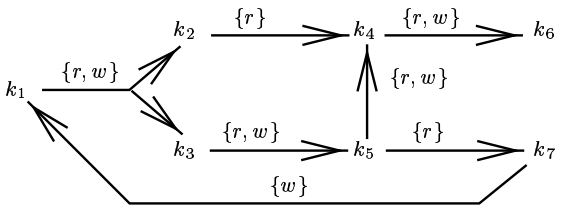

Figure 1. A delegation network

although no access right are actually delegated all the way around the cycle.

The delegation of access rights will be formally defined in Sec. 2.3 but we can give the following intuitive interpretation for the figure. k_1 could, for example, be the key of a database server. In that case, k_2 and k_3 belong to two trustees who can together but not alone authorize access to the server. Key k_4 gets the right to operation read (r) from both trustees and further delegates it to k_6. No other keys have sufficient rights to access the server.

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 an authorization.

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. 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 policy functions for combining them as in [3].

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 probably 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 $o \in auth(c)$, $\langle k_1, c \rangle \in Flow$, 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 The intersection of all relations satisfying Rules 1 and 2 is the smallest such relation. □

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 $authorizes_{DN}(k_1, k_2, o)$ 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. It should be noted that the two paths delegating the right r from k_1 to k_6 partially

overlap. In contrast to models where certificates convey identity instead of authority (e.g. [11, 7]), independence of the delegation paths does not affect the confidence level of the conclusions in our model.

Usually, the idea is that a client is entitled to request an operation from a server if there exists a chain of certificates in which the server itself authorizes the client to the specific operation. In addition to the certificate chain, the server only needs its own authentic public key to verify the access rights.

The idea of minimality in Def. 4 is that all tuples in the relation $authorizes$ should have an explicit reason for being there. A straightforward consequence of the minimality is 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. That is, either $k_1 = k_2$ or there is a certificate issued by k_1 such that all its subjects authorize k_2 for the same operation.

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 6 *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 7 (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.

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 8 *Let DN be a delegation network such that $authorizes_{DN}(k_1, k_2, o)$. 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

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

Corresponding cumulative collections of keys and certificates are

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

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 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 $j \in \{0, 1, 2, \ldots\}$. 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 $i = j-1, j-2, \ldots, 1, 0$ 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, a c like that 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 $i = j-1, j-2, \ldots$, 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, $i = j-1, j-2, \ldots, 1, 0$, $Keys_i$ 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 8. □

The above theorem is consequence of the requirement for the subject set 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. In implementations, we are, however, interested in delegation that depends only on a finite number of certificates.

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.

Next we will prove Theorem 6. 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 8.

Proof of Theorem 6 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 8, 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 $k_1 \neq k_2$, 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 delegation network. Thus, the assumption is false and there is a certificate in DN' and in DN whose only subject is k_2. □

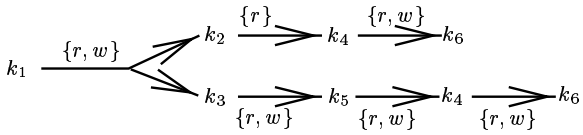

Figure 2. A delegation tree

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 [2].

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 2 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 $\langle Nodes, Arcs \rangle$ is an acyclic directed graph formed by a set of nodes $Nodes$ and arcs $Arcs \subseteq Nodes \times Nodes$ connecting them. If $\langle n, n' \rangle \in Arcs$, 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 $h(Nodes) = \{h(n) \mid n \in Nodes\}$.

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

Definition 9 (homomorphism 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* homomorphism *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* n' *such that* $\langle n, n' \rangle \in Arcs$ *to the keys* k *such that* $\langle h(n), k \rangle \in Flow$.

According to the definition, h is simply a homomorphism 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 homomorphism 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.

Definition 10 (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 homomorphism 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 11 *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 We observed earlier that all leaf nodes map into keys and that the nodes corresponding to keys and certificates alternate. From this, the lemma follows by induction on the depth of the subtrees. □

Finally, we are ready to show that the authorization problem can be formulated as a question on the existence of delegation trees.

Theorem 12 *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 Conditions 1–3 of the theorem are satisfied. Every node of the 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 11 shows that n is mapped into a key. Let n' be a child node of n. A child n' exists because $i + 2 > 0$. The child is mapped into a certificate $h(n')$, and its children into keys. By Lemma 11, 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. 9, 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. 9, 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. Theorem 8 showed 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 $h(n_{-,k_1}) = k_1$. 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 $Arcs_i = \{\langle n, n'_{n,c}\rangle, \langle n'_{n,c}, n'_{n,k}\rangle \mid n \in Nodes_{i-1} \wedge \langle h(n), c\rangle \in Flow \wedge \langle c, k\rangle \in Flow\}$. 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 homomorphism 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 8 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 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. 10) 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.

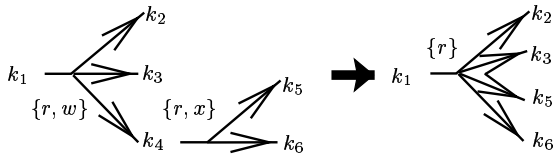

Figure 3. Certificate reduction

4 Certificate reduction

The SPKI draft document [5] 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 an 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. 3 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 second 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 13 (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 certificate 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.

4.2 Soundness and completeness of certificate reduction

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 will be essential in proving the soundness.

Lemma 14 *Let DN' be a delegation network obtained by a single certificate reduction from DN. Then, $authorizes_{DN'} \subseteq authorizes_{DN}$.*

Proof Let 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 12 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 12, $authorizes_{DN}(k_1, k_2, o)$.

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 12. (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. For all the nodes n for which $h'(n) = c'$ in DT', we define $h(n) = c_1$ in DT. Since the subjects of c' are a superset of the subjects of c_1, n has children mapping onto all the subjects of c_1. The children of n that do not map into subjects of c_1 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 12 still hold. Therefore, $authorizes_{DN}(k_1, k_2, o)$.

Hence, the theorem holds in all cases. □

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

Theorem 15 (soundness and completeness of certificate reduction) *Let $DN_0 = \langle Keys, Certs, Auths, Flow, auth \rangle$ be a delegation network. $authorizes_{DN}(k_1, k_2, o)$ 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 14 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 12 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 12 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. □

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 SPKI-type 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 group. A set of certificates is considered valid only if the threshold number of subcertificates with the same group 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 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 transformation.

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 16 (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 17 ($authorizes$ relation) *Let $DN = \langle Keys, Certs, OAuths, Flow, auth\rangle$ be a delegation network where the authorizations are open-threshold-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 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. The equality of issuers and threshold values must also be checked by implementations.

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 18 (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 group of certificates in $Open(DN)$ that is 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, except for grouping 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 19 *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 17 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. 18, the number n of certificates corresponding to c in $Open(DN)$ is equal to the number of subjects of c. This n is also the threshold number visible in the authorizations of all the n the certificates. The n 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. 17 are also met.

On the other hand, assume that in $Open(DN)$ for some $id \in Ids$ and $n \in \mathbb{Z}^+$ there exist at least n 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 $auth'(c) = \langle id, n, a\rangle$ where $o \in a$. This assumption has the meaning that the conditions of Rule 2 of Def. 17 are met. Again, Def. 18 requires that there is at least one certificate in DN with the same issuer k_1 and the same n 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 and removal of certificates in DN can be simulated by addition and removal of certificates by the same issuers in $Open(DN)$. This proves that the transformation $Open$ preserves the functionality of the delegation network.

Theorem 20 *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 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 21 *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*

$$authorizes_{DN_3} = authorizes_{DN_4} \quad \text{and}$$
$$issuers(Certs_4 \setminus Certs_1) = issuers(Certs_3 \setminus Certs_2).$$

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 $auth_3(c) = \langle id, n, a\rangle$ for a certificate $c \in C$, then we add to DN_4 a certificate for every set of n certificates in DN_3 whose identifier is id, threshold value n and issuer the same as that of c. The subject sets of these certificates are formed by the subjects of the n 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 n associated with the identifier (this is a (k, n) scheme with $k < n$), the subsets of size n of the certificate group have been enumerated as groups of size n 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. 17 can be applied. As in the proof of Theorem 19, 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 keys [2]. This means that the transformation does not open any new lines of denial-of-service attack by expiring or revoking of certificates.

It should be noted that Theorems 19–21 and the proofs of this section do not only show properties of our proposed certificate scheme. They can be generally 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 Conclusion

We presented a formal model of access right delegation with certificates. The model made it possible to show the equivalence of different techniques for access control decisions in distributed, key-oriented trust management systems. In particular, we proved the soundness and completeness of certificate reduction with respect to the model. The model can also be used as a basis for development of algorithms for managing certificate databases. Moreover, we suggested a simple way for representing threshold certificates and proved it to have desired functional and security properties.

References

[1] Martín Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Transactions on Programming Languages and Systems*, 15(4):706–734, September 1993.

[2] Tuomas Aura. On the structure of delegation networks, Licentiate's thesis. Technical Report A48, Helsinki University of Technology, Digital Systems laboratory, December 1997.

[3] 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.

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

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

[6] Butler Lampson, Martín Abadi, Michael Burrows, and Edward Wobbler. Authentication in distributed systems: Theory and practice. *ACM Transactions on Computer Systems*, 10(4):265–310, November 1992.

[7] Michael K. Reiter and Stuart G. Stubblebine. Path independence for authentication in large-scale systems. In *Proc. 4th ACM Conference on Computer and Communications Security*, pages 57–66. ACM Press, April 1997.

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

[9] Lawrence Snyder. Formal models of capability-based protection systems. *IEEE Transactions on Computers*, C-30(3):172–181, March 1981.

[10] Edward P. Wobber, Martín Abadi, Michael Burrows, and Butler Lampson. Authentication in the Taos operating system. *ACM Transactions on Computer Systems*, 12(1):3–32, February 1994.

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