Regulated Delegation in Distributed Systems *

Xuhui Ao
Ask Jeeves, Suite 400
1551 South Washington Avenue
Piscataway, NJ 08854
xao@askjeeves.com

Naftaly H. Minsky
Department of Computer Science
Rutgers University, Piscataway, NJ 08854
minsky@cs.rutgers.edu

Abstract

Certificate-based delegation (CBD) is a prominent element of distributed access control, providing it with flexibility and scalability. But despite its elegance and effectiveness, CBD has inherent limitations that restrict its applicability. These limitations include, among others: lack of support for non-monotonic *policies, such as separation of duties; the inability to support the* transfer *of privileges, where the delegator loses the privilege it delegates; and the lack of support for* quotas*, i.e., restrictions on the number of time a given privilege can be exercised.*

This paper describes an approach to the distributed delegation, which shares much of the flexibility and scalability of CBD, but is not encumbered by its limitations. This approach is based on the decentralized control mechanism called law-governed interaction (LGI), which is used to regulate the process of delegation itself.

1 Introduction

Broadly speaking, delegation is an act of granting privileges to others. A *delegation policy* is a specification of such things as who is permitted to delegate which privileges to whom, and under which circumstances, and how such delegation should affect the privileges of the delegator itself. The nature of such policies, and the mechanisms for enforcing them in distributed systems, are the subjects of this paper.

The currently leading approach to distributed delegation has come to be known as "certificate-based delegation," (CBD, for short), which is strongly associated with the, so called, *trust management* approach to authorization, developed under several projects, such as: SPKI/SDSI [6], KeyNote [5], and Delegation Logic [7]. Certificate-based delegation has been described succinctly in [7], as follows:

> *In the "trust-management" approach to distributed authorization, a "requester" submits a request, possibly supported by a set of "credentials" issued by other parties, to an "authorizer," who controls the requested resources. The authorizer then decides whether to authorize this request by answering the "proof-of-compliance" question: "Do these credentials prove that a request complies with my local policy?".*

The credentials mentioned in this description include the so called *delegation certificates*—which are particularly effective when SPKI/SDSI is employed, identifying principals (delegators and delegatees) with their public keys.

Despite the elegance of this approach, and the flexibility and scalability it provides (which have been discussed in a superb study by Aura [3]), CBD has inherent limitations that restrict its applicability. These limitations—most of which are well recognized by the proponents of CBD [3]—include, among others: lack of support for *non-monotonic* policies, such as separation of duties; the inability to support the *transfer* of privileges, where the delegator loses the privilege it delegates; and the lack of support for *quotas*, i.e., restrictions on the number of time a given privilege can be exercised.

We submit that these limitations are due to that fact that CBD imposes no control over the process of delegation. All that matters, according to CBD, is the bundle of certificates submitted to the authorizer which it evaluates according to its own policy. This freewheeling delegation process provides CBD with great flexibility, and scalability. But it also limits its effectiveness, in several ways.

First, certain types of delegation policies have to do with the delegation process itself, and their violations cannot be detected by the authorizer, from the bundle of certificates it gets. A case in point is the requirement that certain rights would be delegated only *by transfer*; that is, the delegator would give up its own right, when delegating it to somebody else. It is, of course, impossible for the authorizer to determine whether or not a delegator did give up its own right.

*Work supported in part by NSF grant No. CCR-04-10485, and by the NJ Commission on Science and Technology "Excellence Award"

Delegation by transfer is, therefore, not supported by CBD, as is well known.

Moreover, the lack of control over the delegation process may be problematic even for such policies where the authorizer can determine the right of the requester, purely on the basis of the bundle of certificates it is presented with. This is for two main reasons. First, as we will demonstrate in the following section, the authorizer may be in no position to examine and evaluate the set of supporting certificates submitted to it by the requester. This may be because the authorizer might not have sufficient knowledge for such an evaluation, or because it does not have the right to even examine such certificates. Finally, the need to verify the validity of a large number of certificates for every service request could be overly time consuming.

As a simple everyday example, note that when accepting a dollar bill as payment for a service, we are, in a sense, getting the right to claim a quantity of gold from the Bank of America—or so it was once. But we accept this privilege without examining the sequence of "delegations" that transferred this bill from the Mint that produced it, all the way to us. Indeed, we would not have the right for such an examination, and it would have been "computationally" impractical to do so. We trust, instead, the delegation process itself. That is we trust (without complete justification, in this case) that this bill has always been moved, from one hand to another, but never copied. No such trust is available under CBD, as we already pointed out.

We conclude, therefore, that the process of delegation itself aught to be regulated, if we are to eliminate the above mentioned limitations of CBD. But it is clear that such regulation must be done in a decentralized manner, if we are to retain the flexibility and scalability of CBD. In this paper we show how such regulation of delegation can be done effectively and scalably via the decentralized control mechanism called *law-governed interaction* (or, LGI). We should point out, however, that the proposed treatment of distributed delegation is not a replacement for CBD, but is complementary to it. In particular, because LGI itself is implicitly based on a restricted form of CBD, and because LGI needs to use CBD explicitly for certain policies.

The rest of this paper is organized as follows: We start with an example that would help us demonstrate the above mentioned limitations of CBD. In Section 3 we provide a very brief introduction to LGI, whose manual is available, along with the implemented mechanism itself, through [10]. In Section 4 we show how the example policy introduced in Section 2 is implemented under LGI; and in Section 5 we reflect on the nature of delegation under LGI, and on its complementarity with CBD. We conclude in Section 6.

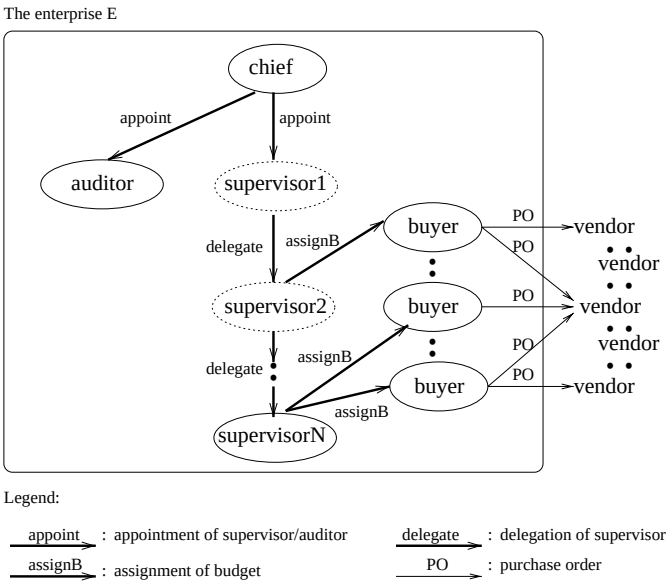

Figure 1. The interaction among different agents, under policy P.

2 On the Limitations of Certificate-Based Delegation

We will attempt here to demonstrate the limitations of CBD via an example. We will show later how this example can be formalized and enforced by LGI, with some use of CBD.

2.1 Regulating Purchasing within an Enterprise: an Example

Consider a community C of distributed agents (people, and software components working on their behalf) belonging to some enterprise E. These agents are partitioned into two disjoint groups, called *management* and *staff*. We are interested in the purchasing activity by the members of the staff (also to be called *buyers*), carried out by sending purchase orders (POs) to various vendors (which are not part of enterprise E). We assume that the enterprise has a policy P that governs the purchasing by members of the staff, and the manner in which such purchasing is to be supervised by certain members of the management group. This policy is specified, informally, below:

Policy P: This policy involves three managerial roles: *chief*, *supervisor* and *auditor* (see Figure 1). Their authority with respect to the buyers, and with respect to each other, is specified below:

1. *There is a single, fixed, chief in community C. It*[1] *can appoint two* different *members of the management group to the roles of supervisor and auditor.*

2. *The chief can supply the supervisor with a purchasing budget, which the supervisor has the right to distribute among buyers—without the right to use this budget for issuing its own POs.*[2]

3. *Each buyer is allowed to issue arbitrary POs, provided that their cumulative cost does not exceed the buyer's budget.*

4. *The supervisor is allowed to delegate its supervisory role to another member of the management group, subject to the following constraints:*

   (a) *This delegation must be* by transfer, *like the passing of a leader's baton, so that the community would have no more than one supervisor at a time.*

   (b) *When delegating its supervisory role, one has to transfer its remaining purchasing budget to the new supervisor.*

   (c) *The role of supervisor should never be delegated to an agent playing the role of an* auditors. *(This is an example of what is known as* separation of duties *constraint).*

   (d) *The chief must be notified of every transfer of the supervisor role.*

2.2 Discussion

The most notable aspect of this policy, as compared to policies under CBD, is that it is *communal*. That is, it is not the policy of any particular agent, with respect to the use of its resources, but it governs the interaction of various agents in community C. Indeed, the main purpose of this policy is to regulate the process of delegation in this community, in order to ensure that certain global, i.e. communal, properties are always satisfied. These include the following properties: (a) the total cost of purchase orders issued by members of this community never exceeds the budget provided by the chief; (b) there can never be more that one supervisor in the community, at the same time; and (c) the same agent cannot play the roles of supervisor and auditor at the same time.

One may attempt to cast this communal policy into the CBD framework, as follows: We will have each vendor adopt a policy of accepting a PO only if it is accompanied with a "proper" chain of delegation certificates. Namely, a chain starting with the certificate issued by the chief appointing his first supervisor; continuing with the sequence of certificates that delegated the supervisory role, from one agent to another; and ending with the certificate signed by the last of these supervisors, providing the requester with its budget. But this attempt has several serious problems:

- By examining the set of certificates presented to it, a vendor would not be able to ascertain that most of the requirements of policy P have been satisfied. In particular, and basically for the reasons explained in [3], the vendor would not be able to ascertain: (a) that there is only one supervisor in the system; (b) that the supervisor does not play the role of auditor as well; and (c) that the requesting buyer still has sufficient funds in its budget, and did not use these funds by POs sent to other vendors.

- The vendors, which do not belong to enterprise E, might not be in a position, or have the right, to see the relevant set of delegation certificates, or to check if they satisfied policy P (even if they were able to do so). The reason is that policy P may be considered a private matter of the enterprise, which it may want to change occasionally without having to inform the various outside vendors it might trade with. Moreover, even if the policy itself is to be known to the vendors, the specific agents involved in a given delegation chain might be considered confidential by the enterprise.

We will see later why these difficulties do not mar our LGI-based implementation of this policy.

3 Law-Governed Interaction (LGI)–a Brief Overview

LGI is a message-exchange mechanism, originally introduced in [8], that allows an *open group* of distributed agents to engage in a mode of interaction *governed* by an explicitly specified policy, called the *law* of the group. The messages thus exchanged under a given law $\mathcal{L}$ are called $\mathcal{L}$-messages, and the group of agents interacting via $\mathcal{L}$-messages is called an $\mathcal{L}$-community, denoted by $\mathcal{C}_{\mathcal{L}}$, or simply by $\mathcal{C}$. LGI has been recently released for free download through `http://www.moses.rutgers.edu/`.

We provide here only a very brief overview of this mechanism, attempting to make the rest of the paper understandable for people not previously familiar with LGI. For more information about LGI, particularly about the structure of its laws, its law enforcement mechanism, its deployment, and its treatment of certificates, the reader is referred to the manual supplied with the release of LGI [10], and to [1], for some features not included with the release.

[1] Since an agent, in this context, may be either a person or a software component, we will refer to agents by "it".

[2] We leave the authority of the auditor unspecified, for simplicity.

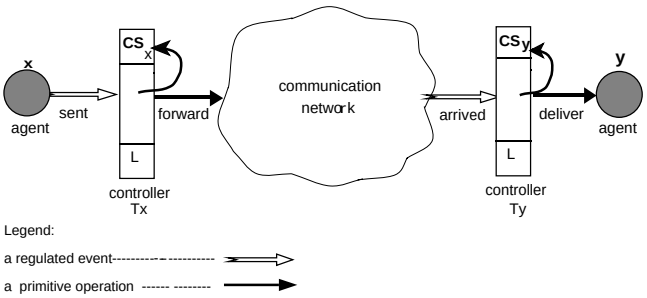

Figure 2. Enforcement of the law.

By the phrase "open group" we mean (a) that the membership of this group (or, community) can change dynamically, and can be very large; and (b) that the members of a given community can be heterogeneous. In fact, we make here no assumptions about the structure and behavior of the agents that are members of a given community $\mathcal{C}_{\mathcal{L}}$, which might be software processes, written in arbitrary languages, or human beings. Both the *clients* and the *servers* of the traditional distributed system terminology are viewed here as agents. All the members are treated as black boxes by LGI, which deals only with the interaction between them via $\mathcal{L}$-messages, ensuring conformance to the law of the community. (Note that members of a community are not prohibited from non-LGI communication across the Internet, or from participation in other LGI-communities.)

For each agent `x` in a given $\mathcal{L}$-community, LGI maintains, what is called, the *control-state* $\mathcal{CS}_x$ of this agent. These control-states, which can change dynamically, subject to law $\mathcal{L}$, enable the law to make distinctions between agents, and to be sensitive to dynamic changes in their state. The semantics of control-states for a given community is defined by its law, could represent such things as the role of an agent in this community, and privileges and tokens it carries.

The nature of LGI laws: An LGI law is defined over a certain types of events occurring at members of a community $\mathcal{C}$ subject to it, mandating the effect that any such event should have. Such a mandate is called the *ruling* of the law for the given event. The events subject to laws, called *regulated events*, include (among others): the *adoption* of the law by the agent, the *sending* and the *arrival* of an $\mathcal{L}$-message. The operations that can be included in the ruling for a given regulated event, called *primitive operations*, are all local with respect to the agent in which the event occurred (called, the "home agent"). They include, operations on the control-state of the home agent and operations on messages, such as `forward`, `deliver` and `release`. Our middleware currently provides two languages for writing laws: Java, and a somewhat restricted version of Prolog. We employ prolog in this paper. In this case, the law is defined by means of a Prolog-like program `L` which, when presented with a goal `e`, representing a regulated-event at a given agent `x`, is evaluated in the context of the control-state of this agent `cs`, producing the list of primitive-operations `r` representing the ruling of the law for this event.

Distributed Law-Enforcement: Broadly speaking, the law $\mathcal{L}$ of community $\mathcal{C}$ is enforced by a set of trusted agents called *controllers*, that mediate the exchange of $\mathcal{L}$-messages between members of $\mathcal{C}$. Every member `x` of $\mathcal{C}$ has a controller $\mathcal{T}_x$ assigned to it ($\mathcal{T}$ here stands for "trusted agent") which maintains the control-state $\mathcal{CS}_x$ of its client `x`. And all these controllers, which are logically placed between the members of $\mathcal{C}$ and the communications medium (as illustrated in Figure 2) carry the *same law* $\mathcal{L}$. Every exchange between a pair of agents `x` and `y` is thus mediated by *their* controllers $\mathcal{T}_x$ and $\mathcal{T}_y$, so that this enforcement is inherently decentralized. Although several agents can share a single controller, if such sharing is desired. (The efficiency of this mechanism, and its scalability, are discussed in [9].)

Controllers are *generic*, and can interpret and enforce any well formed law. A controller operates as an independent process, and it may be placed on any machine, anywhere in the network. We have implemented a *controller-service*, which maintains a set of active controllers. To be effective in a widely distributed enterprise, this set of controllers need to be well dispersed geographically, so that it would be possible to find controllers that are reasonably close to their prospective clients.

Implementation Status: The LGI mechanism has been fully implemented, and released for public use in October 2005 (URL: http://www.moses.rutgers.edu). Space limitations preclude the detailed discussion of this implementation. We will only say that with fairly standard equipment, an LGI controller takes about 100 microseconds for every evaluation of a law of the size of law $\mathcal{P}$, introduced in the following section.

4 A Case Study

We now show how the purchasing policy P, described in Section 2, can be established, in its entirety, via LGI. We start by formalizing policy P into an LGI law $\mathcal{P}$. And we will try to explain this law, attempting to make it understandable even to people not previously familiar with LGI. Then, in Section 4.2, we will amend this law slightly, enhancing the trustworthiness of the POs sent under this law, to the vendors that do not operate under it.

4.1 Law $\mathcal{P}$

Law $\mathcal{P}$ is displayed in Figures 3 and 4. Like any other LGI law, it has two parts: called the *Preamble*, and the *body*.

The preamble of $\mathcal{P}$ has several clauses. The `law` clause indicates its name, and the public key of a certification authority (CA) that is to be used for certifying the controllers interpreting this law. (Note that every law can specify its own CA.) The `authority` clause specifies the public key of a CA—called here `''admin''`—whose certification would be accepted, under this law, for the authentication of agents as employees of certain types of our example enterprise E. Finally, the `alias` clause specifies the address of the distinguished agent `chief`. The body of this law is a list of rules, each of which is followed by informal comments, in italics.

We discuss this law by explaining its treatment of the various activities it regulates, as follows: the *adoption* of this law, an act that makes an agent into a member of the $\mathcal{P}$-community; the appointment of a supervisor, and of auditors, by the chief; the delegation, by transfer, of the supervisory role; the assignment of budgets to buyers, by the supervisor; and the sending of POs to vendors.

(a) Adopting law $\mathcal{P}$: For an agent `x` to adopt law $\mathcal{P}$, thus joining the $\mathcal{P}$-community it needs to choose a generic controller, supply it with the law $\mathcal{P}$ under which it wishes to operate, and possibly supply it a certificate issued by the enterprise administrator–`admin` for the enterprise employees, with the attribute `type(management)` or `type(staff)`. By Rule $\mathcal{R}1$, this would cause either the term `type(management)` or `type(staff)` to be added to its control state for the enterprise employees. Due to the provisions of law $\mathcal{P}$, this term would create a major distinction between these two kinds of employees. For the vendor, it can adopt this law at its will.

We note here that any agent can join this community at any time, possibly by supplying a required type of certificate. And that there is no central authority that regulates the community membership, or that is involved, in any way, in the act of joining it. This is generally true under LGI.

Since we do not compel anybody to adopt and operate under any particular law, or to use LGI, for that matter, how can we be sure that all purchase orders in a given enterprise are issued under law $\mathcal{P}$? The answer is that an agent may be *effectively compelled* to adopt law $\mathcal{P}$ and exchange $\mathcal{P}$-messages, if he needs to use services provided only under this law, or to interact with agents operating under it. For instance, if the vendors accept only $\mathcal{P}$-messages, then anybody needing to issue the purchase orders would be compelled to send $\mathcal{P}$-messages to them. Conversely, if employees make their purchase orders via $\mathcal{P}$-messages, vendors would be compelled to accept such messages, if they are

$\mathcal{P}$*reamble:*

```
law(name(P),ca(publicKey1)).
authority(admin,publicKey2).
alias(chief,"chief@enterprise.com").
```

$\mathcal{R}1$.
```
adopted(Arg)
 :- if (Arg==certificate(issuer(admin),
    subject(Self),
    attributes([type(T)])),
    (T==management|T==staff))
    then do(+type(T)).
```

An employee with the `management` *or* `staff` *certificate issued by* `admin` *will has its type recorded in its CS, once it adopts this law.*

$\mathcal{R}2$.
```
sent(chief,appoint-supervisor(B),
    X) :- not(sAppointed@CS),
    do(+sAppointed),do(forward).
```

Only the `chief` *can appoint the initial supervisor, giving it an arbitrary initial budget* `B`*—but only if currently there is no supervisor.*

$\mathcal{R}3$.
```
arrived(chief,appoint-supervisor(B),
    X) :- if (role(auditor)@CS|
    not(type(management)@CS))
    then do(forward(X,exception(failed-
    delegation(B)),chief))
    else do(+role(supervisor)),
    do(+budget(B)), do(deliver).
```

Acceptance of the appointment to be the supervisor.

$\mathcal{R}4$.
```
sent(chief,appoint-auditor,X)
 :- do(forward).
```

Only the chief can appoint the auditors.

$\mathcal{R}5$.
```
arrived(chief,appoint-auditor,X)
 :- if (role(supervisor)@CS|
    not(type(management)@CS))
    then do(forward(X,
    exception(appoint-auditor),chief))
    else do(+role(auditor)),
    do(deliver).
```

Acceptance of the appointment to be the auditor.

$\mathcal{R}6$.
```
sent(X,delegate-supervisor(B),Y)
 :- role(supervisor)@CS, budget(B)@CS,
    do(-role(supervisor)),
    do(-budget(B)), do(forward).
```

A supervisor can delegate its supervisor's role to others, along with the remaining budget.

Figure 3. Law $\mathcal{P}$

to receive any orders.

(b) Appointment of the supervisor and of auditors: By Rule $\mathcal{R}2$, the `chief` can appoint the initial supervisor by sending the `appoint-supervisor(B)` message where `B` is an arbitrary initial purchasing budget; he can do that only if he did not already appoint one, as evidenced by the `sAppointed` term. The appointment will take place upon the arrival of this message, according to Rule $\mathcal{R}3$, and only if the appointee is a `management` agent and doesn't hold the role of `auditor` (thus observing the separation of duties constraint[3] of this policy). Otherwise, an exception message will be forwarded to the `chief`. Note that the term `role(supervisor)` in the control state of an agent represents that it is currently acting as the supervisor, and the term `budget(B)` indicate that it has `B` amount of purchasing budget, for distribution among the buyers.

Similarly, the `chief` can appoint any `management` type of agent `x` to be an auditor, if `x` doesn't hold the role of `supervisor`, as in Rule $\mathcal{R}4$ and $\mathcal{R}5$. Note, the `chief` can appoint any number of auditors, which is different from the appointment of the supervisor. Also, to simplify our law, we didn't specify the privileges of the agent with the role of `auditor`.

(c) Delegation, by transfer, of the supervisory role: By Rule $\mathcal{R}6$ and $\mathcal{R}7$, a supervisor can delegate its supervisory role to other `management` members by sending the `delegate-supervisor(B)` message, where `B` is its current purchasing budget. The very sending of such a message would remove the supervisory status, and the budget from the sender, thus observing Points 4a and 4b of policy P. When this message arrives at its destination y, it will make y into the new supervisor, with the budget just removed from the sender—this, unless y is not a `management` agent, or if it holds the `auditor` role. In the latter case an exception message will be sent to the chief—and would be received by the chief by Rule $\mathcal{R}8$—allowing the chief to make another appointment of a supervisor.

(d) Assignment of budgets: Only the supervisor can assign the purchasing budget to the buyers by sending message `assign-budget(B1)`, under Rules $\mathcal{R}9$ and $\mathcal{R}10$. If the message receiver is not a staff, then an exception message will be forwarded to the `chief`, otherwise those assigned budget will be represented as the term `budget(B)` in the control state of the buyer.

[3]We note that this provision does not prevent the same person from adopting law $\mathcal{P}$ twice, and then assuming the roles of supervisor and auditor at the same time. This issue can be handled under LGI, at the cost of some complexity, as shown in [1].

$\mathcal{R}7$.
```
arrived(X,delegate-supervisor(B),
    Y) :- if (role(auditor)@CS|
    not(type(management)@CS))
    then do(forward(X,exception(failed-
    delegation(B)),chief))
    else do(+role(supervisor)),
    do(+budget(B)), do(forward(Y,
    delegate-supervisor(X,Y,B),chief)),
    do(deliver).
```
Acceptance of the supervisor delegation, along with the budget.

$\mathcal{R}8$.
```
arrived(X,M,chief)
 :- (if (M==exception(failed-
    delegation(B))) then
    do(-sAppointed)), do(deliver).
```
The arrival of the exception message at the chief.

$\mathcal{R}9$.
```
sent(X,assign-budget(B1),Z)
 :- role(supervisor)@CS,budget(B)@CS,
    B>=B1, do(decr(budget(B),B1)),
    do(forward).
```
Only a supervisor can assign the budget to the buyers, by taking it from its own budget.

$\mathcal{R}10$.
```
arrived(X,assign-budget(B1),Z)
 :- if (not(type(staff)@CS))
    then
    do(forward(Self,exception(assign-
    budget(B1)),chief))
    else if (budget(B)@CS) then
    do(incr(budget(B),B1)) else
    do(+budget(B1)).
```
The buyer accepts the assigned budget from the current supervisor.

$\mathcal{R}11$.
```
sent(X,purchase-order(specs(S),
    payment(P)),V)
 :- type(staff)@CS, budget(B)@CS,B>=P,
    do(decr(budget(B),P)),do(forward).
```
The buyer can issue the purchase order to the vendor if it has enough budget for the payment.

$\mathcal{R}12$.
```
arrived(X,purchase-order(specs(S),
    payment(P)),V) :- do(deliver).
```
The purchase order will be delivered to the vendor without any further ado.

Figure 4. Law $\mathcal{P}$–continued

(e) The Issuance of POs: Finally, by Rule $\mathcal{R}11$, a buyer can issue a purchase order, to any vendor, for a price that is no higher than its current budget, which is decremented appropriately. Note that although the supervisor also has the budget in its control-state, it can't use that budget to issue the purchase order because it doesn't have the `staff` term.

By Rule $\mathcal{R}12$, the arrived purchase order will be delivered to the vendor without further ado. It is important to note that no checking is necessary at the vendor side, because all the vendor should care about is that the PO it gets has been issued properly, according to the law $\mathcal{P}$ that governs the issuance of POs by agents of enterprise E. This, the vendor knows because he gets the PO as a $\mathcal{P}$-message, which means that it has been sent under law $\mathcal{P}$, and is, thus, valid by definition.

4.2 Sending Purchase Orders to Unregulated Vendors

The issue to be addressed in this section is: what do we do if the vendors do not care to operate under LGI, or, perhaps, cannot do so because they have no easy access to properly certified LGI-controllers. We say that such vendors are "unregulated".

To deal with unregulated vendors we change law $\mathcal{P}$ into law $\mathcal{P}'$ by replacing Rule $\mathcal{R}11$ in Figure 4 with Rule $\mathcal{R}11'$ in Figure 5, and by removing Rule $\mathcal{R}12$ of Figure 4. According to Rule $\mathcal{R}11'$, every PO sent by a member x of the $\mathcal{P}'$-community to a vendor v, is sent to it as an unregulated TCP/IP message (this is what the operation $release$ does) that contains the following two certificates: (1) the certificate issued by the CA identified by its public key `publicKey1` (as defined in the `law` clause of law $\mathcal{P}$ in Figure 3), which authenticates the controller $\mathcal{T}_x$ that mediates interactions with agent x; and (2) a certificate issued by the controller $\mathcal{T}_x$, containing the PO and the text of the law $\mathcal{P}$—which, effectively asserts that this PO has been issued legally under law $\mathcal{P}$.

So, the vendor is getting here a chain of certificates, in the sense of CBD. And if the vendor trusts the CA that authenticates the controller, and if it trusts the logic of law $\mathcal{P}'$—which it can examine, because it has been sent along with the PO[4]—then it should be confident that the buyer x does have enough budget to cover the cost of the PO. This is because this law does not allow a buyer to pay for more than it has in its budget.

[4]When the law needs to be private, only the one-way hash of the law would be sent.

$\mathcal{R}11'$.

```
sent(X,purchase-order(specs(S),
    payment(P)),V)
 :- type(staff)@CS,budget(B)@CS,B>=P,
    do(decr(budget(B),P)),
    POcert=certificate(issuer(ThisController),
    subject(Self), attributes([law(ThisLaw),
    purchaseOrder(specs(S),payment(P),vendor(V)),
    ...])), do(release(X,[purchase-order(specs(S),
    payment(P)),createCertificate(POcert),
    C-certificate],V)).
```

The buyer `X` *can send the authorized purchase order to vendor* `V` *with two certificates: A certificate called C-certificate that authenticates the controller $\mathcal{T}_x$, and a certificate signed by the controller, authenticating the PO—it is called here* `POcert`.

Figure 5. The issuing of purchase orders along with certificates

5 The Nature of Delegation under LGI, and its Complementarity with CBD

We are in a position now to reflect on the nature of distributed delegation under LGI, and on its complementary with certificate-based delegation.

5.1 On the Nature of Regulated Delegation, under LGI

We will make here several observations about delegation under LGI, illustrating them via our case study.

1. An LGI law is communal, regulating the entire process of delegation (of various kinds) within the community governed by it. This is contrary to the *server-centric* nature of CBD, which expects the policy to be defined and enforced by a server. The communal nature of LGI provides it with the ability to support properties such as non-monotonicity, delegation by transfer, quotas over the number of times a right can be used, and others that are not supported by traditional CBD.

2. Generally, LGI does not use certificates to represent rights, or to delegate them. The rights of an agent are represented by its local control-state, and their semantics is defined by the law in question. For example, under law $\mathcal{P}$, the right of an agent to be a supervisor is represented by the term `role(supervisor)` in its control-state. Delegations are, therefore, carried out by effecting an appropriate changes in the state of some agents. Thus, under law $\mathcal{P}$, for a supervisor to

delegate its supervisory role to an agent x, it sends x a message which, upon arrival, would add the term `role(supervisor)` to its state.

Note, however, that, as we will point out below, LGI does make some use of certificates, mostly to determine the role of various agents.

3. As a consequence of the above, there is no primitive notion of delegation in LGI mechanism per se, and none is required. There can be many circumstances which would allow one agent to confer some rights on another. Thus, one may have pure delegation, where one agent gives some of its own rights, or all of them, to another—as in the case of transferring the supervisor rule under $\mathcal{P}$. Or one may have a law that allows one agent to provide others with rights it does *not* have for itself. This is the case with the chief, under law $\mathcal{P}$, appointing somebody to the role of supervisor. These two types of "delegation," and others, can represented in a unified manner, once they are defined by the law of the given community.

4. The semantics of delegation is firmly defined by the law at hand, and can be relied upon by delegators and delegatees. For example, when the chief, under law $\mathcal{P}$, provides a budget to the supervisor it appoints, it can be confident that whatever is done with this budget, and among which buyers it would be distributed, it would be the upper limit for the cumulative cost of the purchase orders made by members of this community.

5. The ability of an LGI law to regulate delegation often eliminates the need to carry around the history of delegation, all the way to its source, which under CBD is accomplished by chains of delegation certificates. Two examples, in the context of law $\mathcal{P}$, would illustrate this point.

   First, consider a current supervisor s, who may have obtained his baton at the end of a long sequence of transfers, starting with the original supervisor appointed by the chief. All s needs to keep in its control-state is the term `role(supervisor)`, because there is no way for it to obtain this term, except via a legal sequence of transfers, mandated by this law.

   Second, when sending a PO to the vendor, all one has to do is to send a pair of certificated (see Section 4.2) proving, in effect, that the PO is sent by a bona fide controller, interpreting law $\mathcal{P}$. This should convince the vendor that the buyer has enough funds in its budget to cover the cost of the PO, even though it has no information about the sequence of delegations of budgets, from the chief, all the way to the requesting buyer.

6. Finally, we point out that despite our ability to regulate delegation (and other activities) under LGI, our access-control mechanism is quite open, flexible, and scalable for a wide range of policies. In particular we note that anybody can join the $\mathcal{P}$-community, anytime, just by finding a controller certified by the CA mandated by law $\mathcal{P}$, and by presenting the required certificates signed by the CA called "admin." The scalability of the mechanism is discussed in [9].

5.2 On the Complementarity of LGI and CBD

LGI is not intended to replace CBD, but to complement it. This, in three different senses. First, LGI is itself based on CBD. Namely, each of the controllers, that collectively form the infrastructure of LGI, needs to be authenticated to be trusted. Our example law, $\mathcal{P}$, shows how such authentication is done via a single certificate. But a more sophisticated use of CBD is possible, and indeed, required when laws are organized into hierarchies (see [1]).

Second, LGI provides means for using delegation certificates, and for producing them, in a regulated manner, as follows:

- LGI employs delegation certificates in order to provide distinctive characteristics, or roles, to certain members of a given community. For example, law $\mathcal{P}$ makes a distinction between agents that belong to the staff of the enterprise, and those that belong to its management, on the basis of the certificate they provide. This is a very simple example of such use of certificates. Indeed, like Keynote [4], an LGI-law can require a whole chain of delegation-certificates in order to provide a given agents with certain privileges. For more sophisticated use of delegation certificates, for identifying roles under LGI, the reader is referred to [2].

- In order to communicate with agents that employ CBD, but do not operate under LGI, an LGI agent can *create* delegation certificates, as has been illustrated in Section 4.2.

Finally, the domain of applicability of LGI and CBD are complementary: LGI is intended to be used for communal policies, and for server-centric policies that require control over the process of delegation—neither of which is served well by CBD. CBD, on the other hand, could be the better choice for server-centric policies that do not require control over the delegation process; a good example of such a policy has been introduced in [3].

6 Conclusion

This paper describes an approach to the distributed delegation, which shares much of the flexibility and scalability of certificate-based delegation (CBD), but is not encumbered by its limitations. The proposed treatment of distributed delegation is not a replacement for CBD, but is complementary to it. First, because their range of suitable application domains is complementary. Second, because LGI, on which our approach is based, itself is based on CBD, and uses it explicitly, when appropriate. The LGI mechanism itself has been fully implemented, and released for public use during the October of 2005.

References

[1] X. Ao and N. H. Minsky. Flexible regulation of distributed coalitions. In *LNCS 2808: the Proc. of the European Symposium on Research in Computer Security (ESORICS) 2003*, October 2003. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

[2] X. Ao and N.H. Minsky. On the role of roles: from role-based to role-sensitive access control. In *Proc. of the 9th ACM Symposium on Access Control Models and Technologies, Yorktown Hights, NY, USA*, June 2004. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

[3] Tuomas Aura. Distributed access-rights management with delegation certificates. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Distributed and Mobile Objects*, volume 1603 of *LNCS*, pages 211–235. Springer, 1999.

[4] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The keynote trust-management systems, version 2. ietf rfc 2704. Sep 1999.

[5] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The role of trust management in distributed systems security. *Secure Internet Programming: Issues in Distributed and Mobile Object Systems*, 1603, 1999.

[6] Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in spki/sdsi. *Journal of Computer Security*, pages 285–322, Jan 2001.

[7] Ninghui Li, Benjamin N. Grosof, and Joan Feigenbaum. Delegation logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security (TISSEC)*, pages 128–171, Feb 2003.

[8] N.H. Minsky. The imposition of protocols over open distributed systems. *IEEE Transactions on Software Engineering*, February 1991.

[9] N.H. Minsky and V. Ungureanu. Law-governed interaction: a coordination and control mechanism for heterogeneous distributed systems. *TOSEM, ACM Transactions on Software Engineering and Methodology*, 9(3):273–305, July 2000. (available from `http://www.cs.rutgers.edu/~minsky/pubs.html`).

[10] Constantine Serban and Naftaly Minsky. The lgi web-site (includes the implementation of lgi, and its manual). Technical report, Rutgers University, June 2005. (available at `http://www.moses.rutgers.edu`).