Submission to the European SIGOPS Workshop, 1988

Karen R. Sollins
Laboratory for Computer Science
Massachusetts Institute Technology
545 Technology Square
Cambridge, MA 02139
USA

April 12, 1988

The conflicting goals of autonomy and interdependence have been central to the two areas where I have been doing research, naming services and authentication. In both cases, not only are there questions about the feasibility of mechanisms and algorithms, but more importantly, we must provide flexibility to support the varying policy needs of organizations attempting to cooperate while maintaining their own autonomy as well. My work on authentication is taking place within the scope of the Mercury Project at MIT, while the work on name service access is occurring within the Distributed Systems Architecture Board (DSAB) Task Force on Naming. The DSAB is a joint effort by people from industry, academia, and the government to coalesce the multitude of work being done on distributed systems in the USA. I will sketch out both of these projects and the choices made in each with respect to autonomy versus interdependence.

Within the Mercury Project, remote invocation has taken a primary role. The project has been one of designing and building mechanisms to support program invocation in a heterogeneous environment. This means that the focus is interoperability and cooperation across language, operating system, and hardware bases. The model of communication patterns is that a client will invoke remote services simply and easily without needing to know anything about the implementation details of the service, other than the operations it supports, how to invoke them, and the responses. One implication of this is that the client also should not need to be directly involved in invoking additional services by the original service to complete a task. Cascading of invocations must be supported, but possibly invisible to the client.

This problem of cascading invocations becomes especially complex when the client and services involved have requirements for authentication. Such authentication requirements may arise from a need for accountability, access control, or privacy. Let us consider a brief example. The client, sitting at a workstation, would like to make travel arrangements. This is done by invoking the travel agency service that happens to be an independent organization. The travel agency will make airline and car rental arrangements and needs to be able to bill these back to the accounts payable office at the client's company. The travel agency will charge the accounts payable office directly for the airline reservations, but will hand off to the car rental agency the task of making the car reservation and charging the accounts payable office appropriately. In this set of tasks, the client, travel agency, car rental agency, and accounts payable office wish to cooperate for the duration of the set of transactions, but require, in addition, the ability to limit the scope of their cooperation and accessibility. It is necessary that they depend on each other temporarily, to the extent that authenticity of the participants can be provided to the satisfaction of the other participants. Additionally, each must be able to retain its independence or autonomy.

The functionality we wish to achieve falls into two categories. First, we must be able to authenticate the principals involved; in other works, we must be able to identify them correctly. Second, it must be possible for the participants or transit points, and especially the original client, to specify access constraints or bounds on how the authenticating information can be used. We can identify a set of requirements for a mechanism needed to support this class of communication.

- **Unforgeability:** the degree to which identities are unforgeable is the degree to which those identities can be trusted. The greater the degree of autonomy among the participants, the greater the requirement of unforgeability.
- **Accountability:** it is often necessary, as in our example, for the final destination to know not only the initial source of a request, but also the intermediate participants. For example, it may be that the accounts payable office will only accept charges for cars for an individual if they have passed through the travel agency and therefore can be trusted to be part of a business trip. This allows the accounts payable office to place some amount of accountability for car rentals on the travel agency.
- **Discretionary restriction:** in order to be able to limit use and possible abuse of identities, it is important that not only the original client, but also intermediate transit points be able to include in the authentication information restrictions on the use of it. In the case of the travel plans, the client might set a monetary limit on the cost of the trip, in addition to a time limit after which the trip charges should not be accepted. The travel agency will need to split the monetary limit between itself and the car rental agency, in order that both will be able to charge part of the trip to the accounts payable office.
- **Modularity:** in order to provide the degree of cooperation required to complete the task, the client must know how to invoke the initial service invoked and in the case of authentication, the authentication and discretionary restrictions needed by the service providing the final destination, but beyond that the tenets of modularity, hiding the intervening internal structure and implementation details must be supported. The traveller should not need to know about the private arrangements between the travel agency and car rental agency that form the basis for their pairwise communication and cooperation.
- **Independence:** autonomy is also present in the requirement that cooperative activities should not need, at least in terms of authentication, to be in contact with all the participants simultaneously in order to accomplish the desired task. It is this requirement that implies that only intermittent connectivity is required. Direct participation by the traveller, who may not support multitasking, should not be required in order to verify authenticity at the time that the travel agency or car rental agency contacts the accounts payable office.
- **Combining identity:** as described in the example, it is often the case that not only will a service be acting on behalf of another in invoking a third, but the third one will only grant access when provided with the combination of identities. In many organizations this is provided by requiring countersignatures or multiple keys in combination. It is only the combination of identities that is acceptable.

In order to make a mechanism providing such services practical, it must also be efficient. In particular, not only must the overhead of using the mechanism be kept as low as possible, but also the cost of not using such a mechanism must be even lower, so that the cost is mostly borne when there is an authentication requirement. Within the Mercury Project, we assume that there will be many situations in which authentication or discretionary restrictions will not be required, since many of the applications will be within a domain of trust and tight cooperation.

The proposed mechanism to meet the requirements is known as the *passport.* It is an object that is passed with whatever restrictions may have been added and is unforgeably stamped at each transit point. The figure below depicts a passport among three principals, A, B, and C, and assumes a secret key for each principal. Encryption is used as the basis for unforgeability. The original client A uses its secret key to encrypt the name of the first transit point B, and A's constraints C_A, and appends its own name and the constraints in the clear. Each transit point then re-encrypts the previously encrypted material along with the name of the next transit point and its own constraints. It also appends its own name and constraints

$$\begin{tikzcd}
A \arrow[r, "\{B, C_A\}^{K_A}, A, C_A"] & B \arrow[r, "\{\{B, C_A\}^{K_A}, C, C_B\}^{K_B}, A, C_A, B, C_B"] & C
\end{tikzcd}$$

in the clear. Encryption at each transit point in conjunction with including the following transit point in the encrypted material provides accountability. Discretionary restrictions are present in the form of a set of constraints, designated by C_A and C_B. Modularity is supported because the client provides only the constraints supported by the final destination. If other information is needed between intervening services, those will be provided by the requesting services, and need not be known to the original client. Independence is provided by including the constraints in the clear as well as encrypted for verification. By presenting them unencrypted as well, any transit point that does not require verification from the authentication service can know the constraints without any external communication. There is a tradeoff here between the number of bits transmitted and the requirement for extrernal communication in order to know the constraints. This can be moderated by encrypting a function of the constraints rather than the full representation of them, thus vastly reducing the number of bits transmitted from twice the size of the constraints. In order to provide passports there is an assumption of pairwise authentication between communicating principals. When the authentication and restriction requirements are limited or unnecessary, the latter can be left null or empty and the former can be avoided with the use of a single flag indicating whether authentication is required. If it is, a passport containing minimal information can indicate that the supporting pairwise authentication is adequate.

The current status of this work is that the design is being reported in late April [1] and implementations are now beginning. These will be done in C and Lisp, since at present it is in those two languages that applications are developing within the Mercury Project. The implementations will allow us to verify not only the functionality of passports, but the efficiency as well.

The work of the Task Force on Naming has concentrated on accessing naming services in an environment of autonomous networks. In the expanding internetwork there are growing needs for both cooperation and maintaining the autonomy requirements of administratively and organizationally independent entities. The problem with which we begin our investigation is discovering the identity of people working in other organizations. Many corporations maintain online user directory services, providing the equivalent of online phone (and electronic mail) book lookup. In the case of telephone access, given the name of an employee, it is generally company policy to permit outsiders to learn the direct phone number of that employee, in order to dial directly. On the other hand, few companies will distribute their phone books outside their boundaries. Similar access constraints should be possible electronically as networks become more interconnected. Again, as with authentication, a comfortable compromise between autonomy and interdependence that supports varying policies is needed.

In the case of naming services the work at present is to define a protocol, known as the *Universal Naming Protocol* or UNP, for communicating with name services is being defined. The protocol defines an architecture for communication with name services as well as the syntax for such communication. In addition, not only must base values of the components of the architecture be defined, but additional values in order to make the protocol effective.

The basic framework of the UNP contains three kinds of entities, typed objects, attributes, and directory functions. All objects are typed and it is these typed objects that are to be looked up by requests to name services. There are two primitive types of objects, *type* and *name service*. A type is defined in terms of attributes. The primitive attributes are *attribute* represented by the string "attribute =" and *type* represented by the string "type =". Directory functions are the operations that can be performed by a name service on request and the most primitive of these is *lookup*, which takes as arguments a type and attribute constraints. The syntax is that a message consists of one or more requests for directory functions to be performed; this set of requests takes the form of a program identifying directory functions with arguments of types and attributes to be executed by a name service. A simple syntax consisting of the set of operations and possibly including a request identifier comprises the invocation. The response, either return values or exception, will be identified either with the request identifier or with the request itself. The communication consists solely of single requests (programs) and their responses.

In order to do anything useful, conventions for types, attributes and directory functions must be included as well. For the sake of brevity, I will only suggest a subset of those here. Some important types are: user, host, printer, and other services of various kinds. Important attributes will include: name, electronic mail address, internet domain name, internet address, speed (both in mips for processors and pages per minute for printers), comment, and other attributes reflecting type and quality of services provided by resources. Directory functions fall into four major categories: resolution and lookup functions, administrative functions, control functions, and library functions. Resolution functions determine matches between the attribute constraints, which in their simplest form simply find matches between constraints values supplied and those of objects, but may also include other forms of comparison. Administrative functions include update functions, management of the internal service such as start/restart and optimize the data for particular types of searches, authentication, integrity checking and encryption for privacy. Control functions include such functions as logical operations on the results of resolution functions, traditional database functions such as projection, and conditionals to be used in creating programs. Library functions provide the ability to support "canned" programs, predefined composites of the above functions.

Underlying this protocol definition is the assumption that the transport protocols to be used by UNP are to be specified only by the name services supporting UNP. Requests to a name service might arrive via a TCP or other connection oriented protocol connection or might arrive without prior connection via a datagram protocol. They may even be encapsulated in mail, if the name service is prepared for that. By identifying only the basic architecture required for name service requests and replies, and decoupling this from either the internal models of the clients and name servers or the underlying transport protocols, we are specifying only what is needed for cooperation in the domain of name services with a minimum of impingement on the autonomy of the local service itself or on its administration to set its own policy requirements for access.

The details of this work are being carried forward by Larry Peterson at the University of Arizona and myself. He has done a partial implementation of the protocol and we are writing a specification to presented to the DSAB prior to a meeting of the DSAB and all its task forces together in midsummer. The protocol specification will be distributed prior to the meeting in order to provide the basis for discussion there.

As this report indicates the two areas in which I am putting most of my efforts are tightly coupled to the question of the interplay between autonomy and cooperation. These two projects take slightly different stances on the degree of interdependence required because the degree of cooperation is different. In both cases, communication protocols provide the needed hooks, but policy decisions are left to the end points of the communication. At this workshop, I hope to discuss embedding policy requirements in the supporting networks and protocols in order to improve communication.

References

[1] K. R. Sollins.
Cascaded Authentication.
In *Proceedings of the IEEE Symposium on Security and Privacy 1988*. IEEE, April, 1988.