CASCADED AUTHENTICATION

Karen R. Sollins

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Ma. 02139

Abstract

This paper addresses a problem that has arisen in building distributed systems in which incomplete trust exists and program composition is necessary. The problem is to permit authentication for both access control and accounting when cascading invocations. The problem can be identified as one of providing cascaded authentication. We have developed a mechanism we call *passports* that are passed along with each stage of the cascade and digitally signed at each transition. The information thus signed is that which is critical to the authentication. The contributions of the work are both in recognizing the problem and in devising a solution that is efficient enough to be usable, although there will be some cost associated with such a mechanism.

1. Introduction

Cascaded authentication is the solution to a set of problems that have arisen from confederations of autonomous systems. The problems occur when disparate computer systems are being called upon to cooperate in the absence of complete trust of each other. Furthermore, the systems are utilized in a cascaded fashion, where one invokes a second, which invokes a third, and so on, until the final service is invoked. The particular problem addressed in this paper is providing authentication in this environment given that both accountability and access control may be required.

The paper begins with an example to explain the problem further, and then discusses related work. Section 2 presents the assumptions and goals that must be met by the solution. We can then discuss providing cascaded authentication by means of a mechanism called a passport in Section 3. Since cascaded authentication is based on pairwise, lower level authentication, a typical pairwise authentication mechanism, that is the one used in this project, is described in Section 4. Before concluding, Section 5 describes how passports will be supported in the Mercury System. Section 6 summarizes the work presented here and the contributions of this work.

2. The Problem

In order to understand the problem, we will first consider a hypothetical example of a need for cooperation in the face of incomplete trust. Because of the incomplete trust, access control and accounting are required. Both of these require authentication, the process of validating clients. The example leads to a general statement of the problem, assumptions and goals. The section concludes with a summary of the works that have addressed similar or related problems or parts of our problem.

Consider making travel arrangements. The travel arrangements will be made from an office workstation with a server running at a travel agency. The trip will include air travel, prepaid car rental, and a prepaid stay at a hotel. The company of the traveller has preferences for airlines and car rental agencies, and the hotel is determined by the nature of the meeting. Furthermore, there is only one travel agency from whom the accounts payable office at the traveller's company will accept direct charges, although they will also accept direct charges from car rental agencies and the hotel on behalf of the travel agency. The traveller indicates through the local workstation to the online travel agency service the dates and location of the trip, and the hotel that is required. The travel agency handles the plane reservations but will not deliver tickets until accounts payable has guaranteed to pay for them. Each of the other organizations involved, the car rental agency and the hotel will not accept reservations without prepayment of a percentage of the total reservation. Again the accounts payable department must guarantee payment. The accounts payable department has restrictions as well. They will only accept charges that can be authenticated as having originated from the traveller. In addition, they require verification of the organization requesting payment, especially the travel agency, since they will only accept bills from the one travel agency. In other words, the travel agency needs to be able to act on behalf of the traveller. In addition, it needs to be able to allow the car rental agency and hotel to act on its behalf and in turn on the traveller's behalf. To summarize, the problem is to permit the needed but limited authentication without undue burden on the participants and resources or undue delay.

In order to understand the scope of the problem, it is important to provide a model of the threats of concern here. The discussion here is based on the analysis and terminology presented by Voydock and Kent[1]. The reader is referred to that paper for a full description of the possibilities.

Part of understanding the threat model is identifying our assumptions about the environment. First, there is an assumption that at least some of the communication is transported over LANs, which are easily tapped providing intruders with easy access to network traffic. Hence, the traffic is subject to passive attacks. On the other hand, a second assumption is that the environment is not considered particularly hostile. We must assume that there will be situations in which the contents of messages must be kept private although this may not be the general case. Therefore, in terms of the threat model and passive attacks, there will need to be a mechanism for providing privacy, but the other forms of passive threats, traffic analysis and violation of transmission security are not considered part of our threat model.

In the area of active threats, those that involve threats to

CH2558-5/88/0000/0156$01.00 © 1988 IEEE

authenticity are of utmost concern although several others are of concern as well. Attacks on authenticity can be attempted by either replaying information that was used correctly by someone else or by modifying the messages as they pass on the network. Both are problems and the possibilities of their occurring must be reduced. In addition to modification of authentication information, modification of the message stream can take two other forms considered potential threats in this environment. These are modification of the contents of individual messages and reordering of a stream of messages. If these and privacy are of concern, a chain encryption scheme will address all three at once. Other mechanisms exist for addressing one without the others. A final area where there is often concern about threats is in denial of service. Although that may be considered a possibility, we take the position that it is identified and prevented by the intervention of humans and we are not providing a mechanism to address it in this project.

A requirement for authentication that is not part of the threat model is efficiency. Efficiency takes two forms. First, any mechanisms proposed to achieve authentication must be efficient. This means that the overhead in using them must be reduced as much as possible. Overheads occur in the usage of resources. In the case of communication, expenses can occur in use of processing time, storage, and the communication medium. All must be kept as low as possible. The second form of efficiency is that of minimizing the cost of indicating that one does not want to make use of the authentication or other security mechanisms. Since, in the case of a university or other fairly friendly and cooperative environment, most communication will not require the overhead of complex authentication and security measures, it should cost as little as possible not to use these mechanisms. Of course, this also implies the ability to turn them off. Efficiency is paramount in both its forms, when adding mechanism to communications protocols.

Although authentication, access control and accounting are closely related and mechanisms exist for addressing parts of all three simultaneously (as our does), the focus of this paper is on authentication. In light of the above discussions of the problems and assumptions we can now consider the actual goals for a solution to the problem to be:

- **Unforgeability**: it is important that something be passed to the final destination to be used for access control and accountability and that there be a mechanism not only for trusting that information be tamper-proof, but also that it be verifiable.
- **Accountability**: it is often necessary to be able to track the route of cascaded requests as part of providing access control. Therefore, identification of each participant in the order of their participation is important. For example, it is important to the accounts payable office that the requests from the car rental agency and hotel have come originally from the traveller and thence through the travel agency.
- **Discretionary restriction**: at each transit point the client at that point may want to restrict access privileges of any service further down the route before the final destination. For example, returning to the example of travel arrangements, if the traveller has given permission to charge up to a certain amount for the trip, the travel agency may want to further restrict the amounts that the car rental agency and hotel can charge, keeping the remainder for air travel.
- **Modularity**: it is important, especially in a widely distributed environment, that a client not need to know the internal structure and implementation of the services it invokes. In a situation such as a global network, remote services at autonomous, but loosely cooperating organizations either may be hidden for local security reasons or may change unpredictably. Returning to the traveller's example, the traveller should not need to know whether the travel agency will bill the accounting office for the car and hotel or pass those actions off to the respective organizations.
- **Independence**: one of the advantages of a distributed environment, especially when cascading as described above is available, is that the client need not be available when the request is being acted upon. Therefore, a goal of this project is to permit independent activity, even when authentication is required. In the case of the traveller, the client should not need to be available for verification of requests since wait listing of reservations may cause those to occur at any time. The client may not only go home in the evening expecting the travel agency to continue working on the travel arrangements, but may even turn off the workstation. One should not need to depend on later verification from the originator or any other participant.
- **Combining of identity**: it is often necessary to be acting as a combination of oneself and a client. This was the case of the travel agency requesting payment from the accounts payable office. Only because a request comes from the travel agency on behalf of the traveller might the charges be accepted.

As a preliminary to presenting the mechanism that will meet these goals it is first useful to understand the work that has been done on this subject and related areas. As mentioned above, Voydock and Kent[1] provide the best analysis in the public literature of potential components of threat models. Such a study is always an important part of understanding security requirements. Pairwise authentication, which is an antecedent and supporting mechanism for what will be proposed here, has been investigated by many researchers. Several early works on the three way handshake are by Sunshine[2] and the TCP protocol[3] for initializing reliable connections and Needham and Schroeder[4, 5] for authentication. Otway and Rees[6] have suggested a more symmetric refinement of the protocol for mutual authentication. Either of these will suffice for pairwise

authentication as a supporting mechanism in the work presented here. The work here is based directly on Needham and Schroeder, but incorporates Birrell's ideas of re-authentication as well[7]. Birrell et al.[8] have suggested a mechanism for cascading trust in authentication servers in cases where no single authentication server is trusted by the client and server. It is possible to use such a scheme in conjunction with either of the pairwise authentication schemes mentioned above. For the purposes of this work we will assume that either a single trusted authentication service exists or that cascading as discussed by Birrell et al. is available. Israel and Linden[9] have pointed out the need for accountability in building distributed systems and Karger[10] proposed a mechanism for distributing tickets to be used for proxy login. The work here can be viewed as a generalization of the proxy login mechanism. In the area of authentication, another strong influence on this work was the development of the Kerberos authentication service by Miller and Neuman[11]. Here, although a slightly different handshake protocol was used, the idea of acquiring a collection of tickets with timeouts as first suggested by Denning and Sacco[12] for later use in authentication was paramount. Since the purpose of developing these authentication mechanisms has been to embed them in the Mercury System[13], that project is discussed in Section 5.

3. Cascaded Authentication

As suggested in the example about travel arrangements it is useful in a distributed environment to be able to allow a remote service to act on one's behalf, but retain some degree of control over the actions that the remote service does on one's behalf. This can be generalized to say that A wants B to see that an operation is done on A's behalf with some of A's privileges at C. That activity may be handed off to D or E, before the requests arrive at C and A wants some control over these handoffs, but should not necessarily need to know all the details of these handoffs. We want to capture the idea of handoff and permit A to set bounds on the actions that B can take on A's behalf. In turn, B may want to limit further what D can do on behalf of A and B. Of course, in the end, C will make the final decisions about control of access in light of A's statements about who can act on its behalf.

The mechanism proposed here to solve this problem of handing of limited authentication is called a *passport*. The passport identifies the originator and is digitally signed at each transit point, so that each participating transit point is identifiable. Furthermore, it includes the limits set or further constrained at each transit point. Digital signatures imply encryption. Therefore, in order to give each transit point the information about the constraints set by previous transit points, the constraints are also transmitted in the clear. We will examine the mechanism in more detail and then enumerate how it meets each of the goals set out above.

The two most important parts of a passport are the unforgeable identity of the transit points and the constraints or limits on the actions for which it can be used. The constraints are passed as data and identified by encryption with a key that only the owner (and in the case of a secret key scheme, the authentication or key distribution server) knows. At each stage, a transit point takes the passport it has been given, adds any further constraints or limits it wishes to impose and encrypts this new information along with the encrypted part of the passport it received. This sort of re-encryption is repeated at each stage until the passport arrives at the final service. There the whole package is sent to the authentication server (in the case of a secret key mechanism) for verification by deciphering in the reverse order of the encryption. This is demonstrated in more detail below.

It is important to understand how much trust is required and the cost of a lack of trust. No trust is required, since each transit point is free to verify the passport by sending it to the authentication service. Presumably this requires a remote access, at the cost of a round trip to the authentication service. Therefore modification in transit can be discovered any time it is suspected.

In order to verify the passports, several pieces of information are needed. First, the identity of each transit point must be known. Hence, these must be included in cleartext. Each transit point adds its own name at each stage. In addition, it is important to provide a check that the encrypted material has not been modified. At each stage, the transit point includes in the encrypted material the name of the transit point to which the passport is being sent. When the authentication server deciphers the passport, it has a simple check that the name of the owner of the previous encryption key is in the next set of deciphered data.

There are two other aspects of passports that are important for effectiveness of the design. The first is the information that is passed in the clear, and the second is the inclusion of a nonce in the encrypted material. As mentioned before, each transit point includes its own name in the clear. In addition, each transit point includes its own constraints in the clear for two reasons, efficiency and verification. If the constraints were only included in the encrypted material, then each transit point that needed to know them would be forced to decrypt them, which, in the case of a secret key scheme, would mean invoking the authentication server, even if true verification were not the objective. In addition, including the constraints in the clear provides a further check when verification is done that the information that was signed has not been tampered with. The inclusion of the names in the cleartext does this as well, in addition to identifying to the authentication server which keys to use for decryption.

A nonce is included in the original passport created by the client. It must be the first item encrypted. The reason for this is strictly for security. All the other information that is encrypted is public knowledge and when coming from a particular client is always encrypted with the same key. Therefore, it is susceptible to cleartext attack. Including an unpredictable number that is only used once means that the material that is encrypted is not known text. Furthermore, since it is important to chain the encryptions, all text encrypted after the nonce will be affected by the encryption of the nonce. As we will see below, if this degree of threat is not of interest, it is easy to omit the nonce. It is only needed in the original passport from the client since each succeeding encryption en route will include this original.

Constraints or limits on the use of a passport have been mentioned above. They fall into several categories. They may restrict the cascaded request itself, for example by setting restrictions on the path that can be used, or they may set bounds on the actions that can be taken at the final destination. We can identify several types of constraints on the cascaded request. It may be important to limit the number of hops that a request can make. For example, in the travel agency example, the accounts payable office might have been willing to take requests from any of several travel agencies, but may refuse to allow one to hand a request off to another, so there might have been a limit

of two on the number of hops. Another limit might be on the identity of the transit points. In this case, the accounts payable office would only take requests from certain travel agencies, certain car rental agencies and certain hotels, but this constraint can be separated from the number of hops. It also may be the case that the traveller states that the request must be used within the next 24 hours. This is one mechanism for limiting potential misuse of the passport. Another would be a limit on the number of times that a passport could be used. Each of these constraints can be checked either by transit points along the way or the final destination. Of course, the final server must verify them all.

In addition, there may be constraints on the activities at the server. For example, it may be that a holder of the passport can only perform a subset of all the operations available on the object in question. In other words, the travel agency will have permission to decrement the travel part of the budget for the traveller's account, but will not have permission to look at the balances. In addition, there may be limits on the values of the parameters. There may be an upper limit on the amount that the planned trip can cost. The accounts payable will refuse amounts over certain limits. If the travel agency wants to hand off parts of the trip to the car rental agency and the hotel, it must hand off part of the limit that it has received to each. This is a case in which further restriction plays an important role. Other constraints are certainly imaginable, but these appear to be an important set.

We can now examine the creation and use of passports. Suppose A wants B to act on its behalf in doing something at C. B, in turn will hand off the activity to D, which in turn will make the request of C on A's behalf. Throughout this and the remainder of the paper a secret key system is assumed, although conversion to a public key system would be straightforward, as Needham and Schroeder[4] demonstrated. We will use the following notation:

$\{\}^K$	the material within the brackets is encrypted with key K.
A, B, C, D	names of principals
I_{A_i}	the i'th nonce unique to module A
K_A	a secret key of module A, known only to A and the authentication server
C_A	the constraints set by A in this passport

Therefore, A will pass to B:

$$\{I_{A_i}, B, C_A\}^{K_A}, A, C_A$$

In turn, when B wants to hand the passport off to D, it will send:

$$\{\{I_{A_i}, B, C_A\}^{K_A}, D, C_B\}^{K_B}, A, C_A, B, C_B$$

Now, B has encrypted both A's signed statement and its own constraint information, and then added to the cleartext its own relevant information. Thus when D receives the information, if it trusts B, it need not request that the authentication server verify or decipher the conditions[1], although that can be done at any stage. They are available in the clear. D in turn will need to sign the request as well before sending it to C. This can be done in the same manner as B did. D may also want to add its own conditions, to be interpreted by C. At each stage, it is necessary to trust the sender of a passport. Therefore a passport can only be sent on a connection where pairwise authentication already exists. It is possible to piggy-back sending a passport onto a pairwise authentication protocol; this will be discussed below in Section 4. Because encryption is necessary in order to provide digital signatures in passports, the supporting pairwise authenticated connection need not also encrypt all its data.

A passport can be requested or sent unsolicited at any time, but the passport must come from the client. A passport authenticates the client, but does not authenticate the server, since no information flows from server to client. In addition to changing the identity of the principal or principals on whose behalf an action or series of actions occur at the server, the conditions limiting that action or the conditions of invocation of the requests can be changed as well.

With the passport mechanism in mind it is now valuable to review the goals set out for solving the problem of cascaded authentication. For each goal, the way in which it is achieved will be indicated.

- **Unforgeability**: unforgeability of passports is achieved by use of the private keys of the participants or transit points. Each key is known only to the individual owning participant or principal and the authentication server. Since the authentication server is trusted with this information it can verify the identity of a participant based on that private password.
- **Accountability**: by including the names of the participants along the way, and in particular in the encrypted part of the passport where they cannot be modified, accountability is provided. We are assuming that each participant will check that the name of the transit point from which it receives a passport matches the name included in the cleartext portion of the passport. In addition, we assume that each point encrypts using its private key. In the case of a single node not doing this, it will be detected by either the next transit point down the line, or at worst during verification. In the case of two or more transit points working in collusion, not much can be done with this mechanism other than being sure that the first correctly operating transit point receives a correctly identified and encrypted passport. There may be stages in between that are not identified.
- **Discretionary restriction**: the ability to include constraints or limitations at each transit point provides for cascaded discretionary restrictions.
- **Modularity**: by superimposing signatures on the passport in such a way that verification is done only at the end, modularity is achieved.

[1] It should be noted that verification of a digital signature in a secret key system, as described by Needham and Schroeder[4] requires accessing the authentication server, so it should not be done lightly. In a public key system it involves decryption with the public key. Either case will probably require a network access.

Each transit point needs to know only about the one to which it is sending the passport and does not have to know about how that transit point will achieve its job.

- **Independence**: again superimposing signatures and permitting such constraints as time limits on a passport's validity allow for independent operation. Since verification can be held off until the final destination is reached and at that time only the authentication server needs to be involved, independence is achieved. As noted earlier, the less trust exists, the more likely it will be that the authentication server will be asked to verify a paaport at transit points.
- **Combining of identity**: by cascading and superimposing signatures it is possible to identify all the participants in a request thereby allowing the server to accept requests only when the required combination of principals is involved.

In summary, at this point we have considered most of the issues surrounding authentication required in cascaded invocation. We began with an intuitive understanding of the sorts of cases in which the problems arise, for example the case of the individual attempting to make travel arrangements including airline tickets and prepayment for other reservations. We then set the specific goals of the project within the constraints of our basic assumptions of the environment, including an analysis of the threat model. The solution was one of a mechanism called *passports* that can be unforgeably stamped or signed at transit points along the way. This depended on pairwise authentication of the participants, so that will be addressed briefly in the next section, followed by a short discussion of the the application of this work to an existing heterogeneous distributed system, in particular, the Mercury System at MIT.

4. Pairwise communication

As mentioned above in Section 3, the passport mechanism depends on the pairs of transit points that are exchanging passports being able to trust the identity of the transit points from which they receive passports. In addition it would be useful for the sender of a passport to trust the identity of the recipient. If each receiver trusts the sender of a passport, at each stage the transitions can be checked at the receiver for the validity of the path. Providing trusted authentication in both directions will avoid some amount of work.

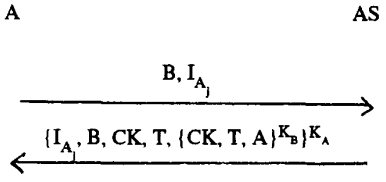
There are numerous examples in the literature of pairwise authentication mechanisms, most notably the work of Needham and Schroeder[4] that addresses this problem using both secret key and public key mechanisms. In the case of a secret key mechanism, four messages must pass between the two principals as well as two preliminary messages between one of the clients and the authentication server. Otway and Rees[6] have proposed a mechanism that requires four messages only for each pairwise authentication, but the problem is that it requires access to the authentication server for each pairwise authentication. It is possible, as has been shown by Miller and Neuman[11] in Kerberos, to batch the requests to the authentication server, and then provide timeouts on the tickets thus acquired. Therefore, the authentication server need not be involved in each pairwise authentication. By doing this, and permitting reuse of tickets when re-authentication is required, the average number of messages in a scheme similar to Needham and Schroeder's can be brought arbitrarily close to four. Since the focus of this work is on cascaded authentication and not pairwise authentication, we will choose to use a scheme similar to Needham and Schroeder's, as well as the three-way handshake of TCP[3] and Denning and Sacco[12]. This will allow for more independence from the authentication server, since it need not be available whenever authentication is needed.

Besides to the requirement of bilateral authentication, there is an additional requirement that an authentication server be provided that is trusted by both principals, or at a minimum there be one that each trusts and they trust each other. This can be extended to say that the principals involved must trust authentication servers that fall into a single closure of authentication servers that trust each other. Birrell et al.[8] have demonstrated an algorithm for achieving this. For simplicity a single authentication server will be assumed here.

In addition to the notation used in Section 3, the following will be used to outline a possible protocol for pairwise authentication.

T	a time stamp
CK	a conversation key generated by the authentication server and exchanged by to authenticating principals
AS	the authentication server

Communication will be assumed to occur instantaneously and lines with arrows will indicate the direction of flow of a message. Time moves down the page.

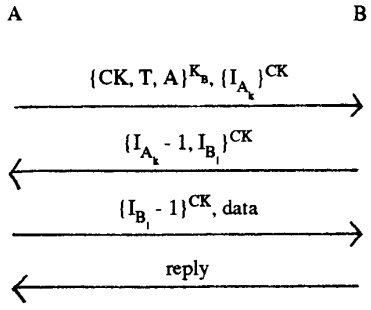


For principal A to acquire a ticket to talk with another principal B and an encryption key to be used for that purpose requires two messages. The first is sent in the clear. There is no need for encryption here, since the only information is the identity of the principal B and a nonce I_{A_j}. The nonce is a number that is unique and provides the basis for a challenge of the authentication server by A. The authentication server meets the challenge by returning to A a message that is encrypted with A's private key K_A (known only to A and the authentication server) containing I_{A_j}, (proving that the authentication server knew A's key), B's identity, CK the key for communication between A and B, T the time to live for that key, and an unforgeable ticket that A can later hand to B. The ticket is unforgeable because it is encrypted with the secret key known only to the authentication server and B. The ticket is used by sending it to B which will use K_B to decrypt it and will find A's name inside; this could only have been done by the authentication server. Further verification of A by B will be seen below. This protocol allows A to verify the response to its request as coming from the authentication server and provides a secret short-lived key for authentication and encryption with B. The fact that the keys are short-lived is part of the

mechanism for preventing the forms of false authenticity identified by Voydock and Kent. If a key is compromised or stolen, its period of validity is limited. Authentication and key exchange for encryption can be piggy-backed on connection initialization, but can also be repeated at later times if the information has been deleted or needs to be changed. The reasons for repetition or modification are related to permitting connections to quiesce or be used on behalf of other principals. These reasons for repeated authentication will be discussed further in Sections 5.

Efficiencies can be achieved here in two ways. The first is suggested by Kerberos, in which requests to Kerberos can be batched. Therefore, rather than incurring the expense of two messages and two encryptions and one decryption[2] (the ticket is not decrypted here) for each connection created by A, there will be two messages and 2N+1 encryptions and 1 decryption for N connections created by A. Of course this can only be done when A can predict the N connections. There is a tradeoff in acquiring a set of keys, because they come with timeouts. If they are not used within their periods of validity, there is a wasted cost of transmission of them. The second form of efficiency occurs if the authentication server and the location registry are the same, allowing requests for finding principals to include the acquisition of the encryption key and ticket for a connection to each module. Such efficiencies become important during usage of such mechanisms.

Once the ticket and encryption key are known, A can initiate an authenticated connection with B, as shown here.



A opens the communication by sending the authenticated ticket previously acquired from the authentication server and a nonce encrypted with CK to B. B uses K_B to decrypt the ticket, verifying that this ticket was supposed to have come from A, and learning CK and T the timeout for it. CK is then used to decrypt the nonce. Both this nonce and I_{B_l} the nonce sent by B to A are challenges. The way the challenges are met is by the recipient decrypting the challenge, modifying it in a known way (in this case subtracting one), and returning the modified version re-encrypted. Only if the key is known by the recipient can this be done correctly, and therefore this proves the identity of the recipient. Neither A nor B will send real information, the data or reply, until its challenge of the other has been met successfully. So A only sends its data in the third message, since B replied successfully to the challenge in the second. If A also includes its reply to B's challenge in the third message, then B can reply to the data in the fourth message. The challenges and responses using the nonces directly counteract the threats of "playback" and "false identity". Passing a passport can occur once A trusts B; therefore the passport can also be sent in the third message of the algorithm along with the data. In fact, it must be sent before the data since it implies a possibility of changing the identity of the principal on whose behalf the information in the data is to be used.

Attacks on integrity, ordering of messages and release of message contents can be prevented by use of encryption of the message data and replies. Since a conversation key has been exchanged this is possible without the threat of compromising a long-lived key, but involves encryption of everything on the connection. The basis for that encryption, the seed for the chained encryption, is the same key CK, known by A and B.

The cost of creating an authenticated connection must be counted in terms of both additional messages and number of encryptions and decryptions. The protocol laid out above requires two additional messages beyond the data and reply messages, plus an average of, at most, the one additional message for accessing the authentication server as described above. There are 7 encryptions/decryptions (encryption of the ticket is done in the authentication server and not part of this protocol) in the handshake, and an average of at most one and a half in accessing the authentication server. If the connection requires encryption of data there is the additional expense of that encryption as well.

It is important to address the effects on authentication of special or exceptional conditions on the connection. This will be considered in Section 5 once the Mercury System itself has been reviewed.

To summarize, for pairwise connections, authentication can be provided by mechanisms similar to those of Needham and Schroeder or Otway and Rees. We have chosen a scheme approximating the former in order to avoid interactions with the authentication server at the time of each authentication. This pairwise authentication is necessary as part of the support for cascaded authentication as we have realized it in passports.

5. Design of an implementation

The purpose of designing authentication mechanisms is to permit use of them. Therefore, it is valuable to understand the usage of passports and their supporting mechanisms in the sort of distributed system for which they were designed. They were designed specifically to solve authentication problems in the Mercury System, a heterogeneous system at MIT.

The Mercury System has two high level goals of importance to this piece of the project, program composition and heterogeneity. Program composition means that it should be possible within the system to invoke programs or services across language boundaries. The second goal is supporting the existing heterogeneity inherent in the choice of languages, operating systems, and hardware. The combination of these two has led to a commitment to providing a common underlying communications semantics that is reflected in each language as an extension to that language. These extensions are known as the language veneers.

The underlying model has two parts that are relevant to authentication and access control; the model of active entities and the semantics of invocation. It is necessary for authentication to understand the nature of a principal in Mercury. Figure 1 depicts a layered model of Mercury. A *module* is a principal, the unit of authentication. Mercury

[2] An encryption costs the same as a decryption in the DES.

consists of modules each running at a single location, although there may be more than one module at a location. A module can be thought of as a service or program running at a specific location, having remotely invocable procedures or *ports*. In addition, a module contains one or more threads of control, known as *agents*. A remote invocation is done by an agent in one module to a port in another. Furthermore, invocations on a group of ports for a single module from a single agent can be sequenced. This is known as a *stream*. A set of streams between a pair of modules can share a *connection*. Thus a module is a principal and resides at a single node. It provides ports and supports a collection of agents. Messages flow between two modules on a bidirectional connection, and a connection can support a collection of one-way streams between agents and sets of ports.

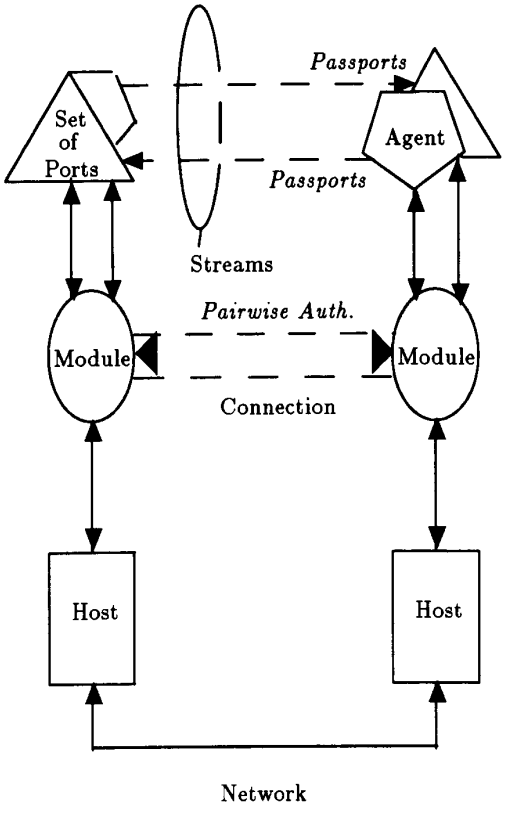


Figure 1: Communication in Mercury

The usage of pairwise authentication and then passports on top are mapped onto the Mercury model of computation as follows. A connection will support pairwise authentication, because each module is a principal and a connection is the means of communication for a pair of modules. When an agent invokes a port it will do so as a component of its owning module, unless it has been given a passport to act on some other module's behalf. The agent will be using a stream for invocation of a set of ports. If the stream is to be used on behalf of another module, the agent will hand a passport to the receiver. The passport will then be handled as described in Section 3. It may be verified or handed along further. If the stream is to be used on behalf of another module only temporarily, yet another passport will be passed along later. If the stream is to be returned to being used on behalf of the original module, a simplified form of a passport can be used, essentially null, indicating a reversion to the original state.

We can now consider means of providing greater efficiency, in addition to those means already mentioned of piggy-backing a passport on connection level authentication and the null passport indicating a return to the default state of authentication. Two other mechanisms to improve efficiency are possible in creating passports, flags indicating the presence or absence of particular constraints and hashing the constraint values. Since every constraint is optional, we will use a one bit flag for each constraint identified in Section 3 indicating its presence or absence. Thus indicating no constraints requires one bit for each potential constraint. If constraints are present, using a pre-arranged hashing function on the constraints in both the encrypted and cleartext forms again permits a reduction of the number of transmitted bits. Thus we can reduce to a small number of bits the ability to not use authentication, and reduce the number of bits for the contraints by hashing.

There are three special conditions that must be considered when building the pairwise authentication mechanisms: connection quiescence, connection exception of failure, and authentication or encryption failure. First, Mercury permits a connection to quiesce, thus allowing information about it to be forgotten by one or both ends of the connection. In this case, re-authentication must be done when the connection is re-established. This is not as simple as it may seem, because, although a connection is bidirectional, it is not completely symmetric since one end or the other acquired a ticket from the authentication server in order to initiate the authenticated connection. There are several possible solutions to this problem. One is that the origianl initiator of the connection acquired two tickets, one for each direction, that have not been forgotten. In this case, either end may re-initialize the connection. A second is that the original holder of the single ticket still has it and re-initializes the connection, assuming it knows to do so. If both of these fail, a new ticket must be acquired. In any case, this is not a difficult problem, but must be addressed.

The second special condition that must be addressed is that of handling authentication and encryption in the face of an exception on or a failure of the connection. When an exception occurs at the connection level, causing the connection to break, re-synchronization will be needed if chain encryption was being used on the connection. In addition, if either end distrusts the authenticity of the other, re-authentication may be required as well.

The third and final exceptional condition that must be considered is the failure of authentication or encryption, and how that will affect the connection on which it is occurring. If there is some failure in either authenticating or the cryptographic activities, this must have the effect of breaking the connection. The reason for this is that the authentication and encryption are begin performed only because one or both participating modules require it. If that cannot happen, the connection should not be used. Once valid authentication can be re-established, the connection can be reused.

We must also understand what happens under three interesting conditions on streams: the stream breaks,

authentication/authorization fails, and the stream quiesces. First, if the stream breaks for some reason unrelated to authentication and authorization, but the state of the stream is not forgotten, nothing needs to be done in terms of authentication in order to restart the stream. In contrast, if authentication or authorization fails, the stream must be put into the broken state because succeeding operations should fail, until the authentication or authorization problem has been resolved. Therefore, once the stream has reached the broken state due to a problem with authentication, in order to restart, a valid passport must be sent to the server. To avoid the need for two types of broken state, one for a security failure and one for everything else, we require that a valid passport always be sent on restarting a stream. (It should be noted that a valid passport can be the null passport representing the module in which the agent resides.) The final issue is what to do if the stream quiesces and therefore can forget its state. In this case, since the principal may have been changed and may not be agent's own module's principal and that information can be forgotten, when the stream is restarted a passport will have to be sent as well. Since all stream information can be forgotten and since, for a particular stream, several passports may be used at different times, passports should be stored somewhere besides only with stream state.

6. Conclusion

In this paper we have investigated the problem of cascading and combining authentication as invocations of remote services are cascaded. The problem can be partitioned, so that pairwise authentication is provided first and the cascading can be done on top of that. Furthermore, we have demonstrated that efficiency is possible where needed, although it will always be the case that if encryption is required, there will be costs associated with it. The mechanisms proposed for achieving these kinds of authentication are based largely on the work of Needham and Schroeder, Birrell, and Karger. The contributions of this work are twofold. The first is in the recognition that a problem has arisen from the combination of distributed systems and incomplete trust among the components of such a system. The problem is that of cascaded authentication. The second contribution is in the generalization of the ideas of capabilities and tickets for proxy login here called *passports* that are a solution to the problem of cascaded authentication.

References

1. V. L. Voydock and S. T. Kent, "Security Mechanisms in High-Level Network Protocols", *Computing Surveys*,Vol. 15, No. 2, June 1983, pp. 135-171.

2. C. S. Sunshine, "Interprocess Communication Protocols for Computer Networks", Tech. report 105, Digital Systems Laboratory, Stanford University, December 1975, This was also submitted as a doctoral thesis.

3. DDN Network Information Center, "Transmission Control Protocol", in *DOD Military Standard Protocols*, E.J. Feinler et al., eds., DDN Network Information Center, SRI International, Menlo Park, CA 94025, DDN Protocol Handbook, Vol. 1, 1985, pp. 1-147 - 1-324, ch. 6.2, The protocol is MIL-STD 1778. This work is also reported in NIC RFC 793, edited by Jon Postel at USC Information Sciences Institute.

4. R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers", *CACM*,Vol. 21, No. 12, December 1978, pp. 993-998.

5. R. M. Needham and M. D. Schroeder, "Authentication Revisited", *Operating Systems Review*,Vol. 21, No. 1, January 1987, pp. 7.

6. D. Otway and O. Rees, "Efficient and Timely Authentication", *Operating Systems Review*,Vol. 21, No. 1, January 1987, pp. 8-10.

7. A. D. Birrell, "Secure Communication Using Remote Procedure Calls", CSL-TR 84-2, Xerox Corporation, Palo Alto Research Center, September 1984.

8. A. D. Birrell, B. W. Lampson, R. M. Needham, M. D. Schroeder, "A Global Authentication Service without Global Trust", *Proc. of the 1986 IEEE Symposium on Security and Privacy*, IEEE Computer Society Press, Oakland, CA, April 7-9 1986, pp. 223-230.

9. J. E. Israel and T. A. Linden, "Authentication in Office System Internetworks", *ACM Transactions on Office Information Systems*,Vol. 1, No. 3, July 1983, pp. 193-210.

10. P. A. Karger, "Authentication and Discretionary Access Control in Computer Networks", *Computer Networks and ISDN Systems*,Vol. 10, No. 1, January 1986, pp. 27-37, This paper was reprinted in *Computers and Security*, 5 (1986), 314-324

11. S. P. Miller and C. Neuman, "Kerberos: Project Athena Technical Plan -- Authentication", Release 1.0

12. D. E. Denning and G. M. Sacco, "Timestamps in Key Distribution Protocols", *CACM*,Vol. 24, No. 8, August 1981, pp. 533-536.

13. B. Liskov, T. Bloom, D. Gifford, R. Scheifler, W. Weihl, "Communication in the Mercury System", *Hawaii International Conference on System Science*, University of Hawaii, Kailua-Koni, Hawaii, January 5-8 1988.