Secure Communication in Distributed Ada

Jörg Kienzle, Thomas Wolf, Alfred Strohmeier
Swiss Federal Institute of Technology
Software Engineering Laboratory
1015 Lausanne Ecublens
Switzerland
email: {jkienzle, wolf, strohmeier}@di.epfl.ch
phone: +41 (21) 693.42.37 – fax: +41 (21) 693.50.79

Abstract. This document describes an implementation of secure communication for distributed applications in Ada 95. It explains the basics of cryptography and introduces the problem of key exchange. After a brief description of distributed systems in Ada 95, the modular structure of the implementation and the way it handles key distribution is presented.

Keywords. Distributed Systems, Security, Ada 95, Object-Oriented Programming, Communication, Cryptography.

1 Introduction

The advent of low-cost, yet powerful workstations combined with fast networks has made the distributed execution of applications over a network a viable alternative to the use of one centralized large computer. To fully exploit the resources of such a distributed system, software has to be designed and implemented in a radically different way than for a centralized system.

The main differences are the inherent parallelism of distributed execution and the fact that the different parts of a distributed application have to communicate with each other to accomplish their task [10]. Various programming models on different levels of abstraction are used to deal with these issues.

Examples for communication paradigms are simple message passing, or on a higher level remote procedure calls (RPC) [2] or remote method invocation. Whatever way is used, potentially confidential data has to be transported from one node of the distributed system to another one. Data travelling over a network can be eavesdropped upon without great effort, provided that the attacker has physical access to the communication lines. One solution to this problem is to encrypt sensitive data.

This paper describes an implementation of secure communication for distributed systems in Ada 95. It is an open implementation in the sense that it allows any cryptographic algorithm to be used.

The outline of this paper is as follows: in section 2 we give a short review of private and public key algorithms, section 3 gives an introduction to distributed systems in Ada 95, section 4 shows different levels of using encryption, followed by an overview of the implementation of the distributed systems annex of the Ada 95 reference manual that we worked with in section 5, and section 6 explains our approach of providing user-transparent secure communication for this implementation.
2 Private and Public Key Algorithms

In modern cryptography plaintext is transformed into ciphertext using an encryption function $E$ and a key $k$ (see fig. 1). The inverse function $D$ expresses decryption of the ciphertext to reproduce the plaintext, again using the same key $k$. This kind of algorithm is called symmetric or private key algorithm [3]. The Data Encryption Algorithm (DEA, [1]) for example is a very popular cryptographic algorithm of this category.

![Fig. 1 Encryption and decryption](image)

The key is an essential feature of a cipher. Keeping the functions $E$ and $D$ secret would suffice to conceal the plaintext, even without a key. But there are two major problems in this case: if the deciphering function $D$ becomes known to a third party, all subsequent communications using $E$ can also be read by that third party; and an encryption function $E$ not using a key is more vulnerable to cryptanalytic attacks than one that does employ a key. Parametrizing the functions with a key $k$ actually defines a class of ciphers: if one key is compromised, it can be changed, while the cipher functions $E$ and $D$ can remain in use.

When encryption is used in communication, there must be prior agreement between the communicating parties about all aspects of the procedure. An algorithm and the method of using it must be agreed upon. The most difficult requirement is that a key must be chosen and made available at both ends of the communication path. Before encrypted data can flow over the line, the key must make a similar journey. Keys can be encrypted using other keys, but in the end at least one key has to be exchanged by some means.

This problem can be solved using public key ciphers, also called asymmetric ciphers, e.g. RSA [8]. Although these cannot be implemented as efficiently as private key ciphers, they can be used to exchange small amounts of data, e.g. a private key that is to be used for subsequent communication.

In public key algorithms, the keys used for encrypting and decrypting are not the same. The decryption key $kd$ is kept secret by the receiver. The other key $ke$ is made public, enabling anyone to encrypt data for the one receiver to whom that key belongs. When $A$ wants to establish a secure communication with $B$, all he has to do is create a private key $k_A$, encrypt it with the public key $ke_B$ of $B$ and send it to $B$ (see fig. 2). $B$ is the only one who knows the decryption key $kd_B$ and therefore the only one who can successfully decrypt the message to obtain $k_A$. Now, both $A$ and $B$ know $k_A$ and they can exchange data using an efficient private key algorithm. This method is used for key exchange in the cryptography implementation for Ada 95 presented in this document.