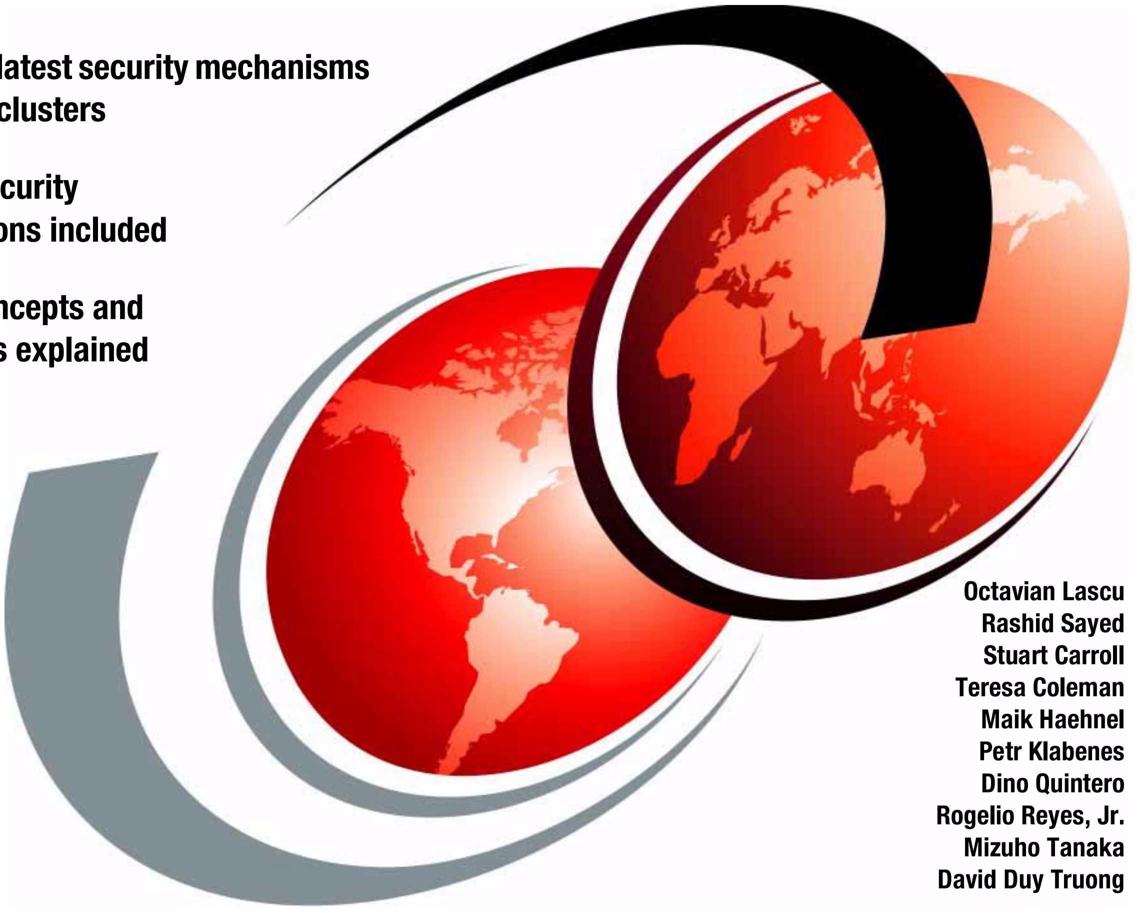
IBM

An Introduction to Security in a CSM 1.3 for AIX 5L Environment

Peek at the latest security mechanisms for pSeries clusters

Practical security considerations included

Security concepts and components explained



Octavian Lascu
Rashid Sayed
Stuart Carroll
Teresa Coleman
Maik Haehnel
Petr Klabenes
Dino Quintero
Rogelio Reyes, Jr.
Mizuho Tanaka
David Duy Truong

Redbooks

ibm.com/redbooks

International Technical Support Organization

An Introduction to Security in a CSM 1.3 for AIX 5L Environment

December 2002

SG24-6873-00

Note: Before using this information and the product it supports, read the information in "Notices" on page ix.

First Edition (December 2002)

This edition applies to Version 1, Release 3, of IBM Cluster Systems Management for use with the AIX operating system Version 5, Release 2.

Contents

Figures

Notices

Trademarks

Preface

This IBM Redbook contains information about the first official release of the new clustering software IBM Cluster Systems Management (CSM) on AIX 5L Version 5.2. Features include base cluster configuration and management, Resource Monitoring and Control (RMC), subsystem access control list setup for shipped CSM resource managers (RM), hardware control, configuration file management, distributed command execution, and a distributed GUI based on the AIX WebSM infrastructure. Included in this release of CSM is a complete set of base security functions based on IBM host-based authentication (HBA) and offered through an abstraction layer in the CSM software. CSM automatically configures HBA for use by the cluster services and establishes secure cluster communications for the shipped CSM resource managers.

This publication is divided in six chapters. The first three chapters are conceptual and include an introduction to security for CSM 1.3 for AIX 5L, security concepts and components, and CSM security infrastructure. Chapter 4 provides practical security considerations. Topics, such as network considerations, security in an heterogeneous environment, and security considerations for hardware control, are discussed. The last two chapters detail secure remote command execution, as well as security administration. Among the topics covered are remote command execution software, OpenSSH installation, and administration of RMC. In addition, we include references to manuals and IBM Redbook publications for reference on the latest information about security for CSM 1.3 for AIX 5L.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

Octavian Lascu is a Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of pSeries clusters and Linux. Before joining the ITSO, Octavian worked in IBM Global Services Romania as SW and HW Services Manager. He holds a Master's Degree in Electronic Engineering from Polytechnical Institute in Bucharest and is also an IBM Certified Advanced Technical Expert in AIX/PSSP/HACMP. He has worked with IBM since 1992.

Rashid Ahmed is a Senior Technical Executive in IBM India. He has nine years of experience in the IT field. He holds a degree in Electronics Engineering from Bombay University. His areas of expertise include HACMP, PSSP, and Sequent systems.

Stuart Carroll is an Advisory pSeries sales specialist for Techline in Bethesda, MD, U.S. He holds a Bachelor of Science degree in Chemical Engineering from Cornell University. He started his career in IBM as a manufacturing engineer, but moved into marketing and has been an RS/6000 pSeries specialist for 10 years. He specializes in large systems (SPs) and clusters and is a subject matter expert (SME) for Regattas p670/690s.

Teresa Coleman is an I/T Architect for IBM Global Services in Ireland. Formerly, she was an MVS and CICS Systems Programmer. She has been working with AIX for eight years. She holds a degree in Computer Science from Trinity College, Dublin.

Maik Haehnel is an I/T Specialist for IBM PreSales Technical Support in Germany. He has three years of experience in clusters on pSeries and xSeries. His areas of expertise include CSM on Linux, AIX on pSeries, and HACMP. He holds a degree in Commercial Information Technology.

Petr Klabenes is a Systems Engineer at GC System a.s. in the Czech Republic. He works as an Advisory IT Specialist supporting pSeries systems. His areas of expertise include enterprise backup and recovery and pSeries clustering.

Dino Quintero is a Project Leader at the ITSO, Poughkeepsie Center. He currently concentrates on pSeries clustering technologies by writing Redbooks and teaching workshops.

Rogelio Reyes, Jr. is a System Services Representative at IBM Philippines. He holds a Bachelor of Science Degree in Electronics and Communications Engineering from Mapua Institute of Technology. He has three years experience in supporting pSeries, RS/6000 servers. His areas of expertise include AIX, PSSP, and storage solutions, mainly ESS.

Mizuho Tanaka is a pSeries IT Specialist at IBM Japan Systems Engineering Co., Ltd. She has been working in technical support mainly for RS/6000 SP and pSeries cluster systems since 1998. She holds a MS in Algebraic Geometry from Meiji University.

David Duy Truong is a Staff Software Engineer at IBM in Dallas, Texas. He has been working in the AIX Supportline since 1997. He holds a Bachelor of Arts in Marketing from the University of North Texas. He is a specialist in PSSP, HACMP, and CATE.

Thanks to the following people for their contributions to this project:

Serban Maerean
IBM Poughkeepsie

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Introduction

This IBM Redbook provides an overview of the security environment in a management domain cluster running IBM Cluster Systems Management (CSM) 1.3 for AIX 5L. We discuss the components and mechanisms that make the cluster secure and the options for changing security methods. For a detailed discussion of CSM 1.3 for AIX 5L installation configuration and management, refer to *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859.

In this chapter, we introduce security topics used in the IBM Cluster Systems Management for IBM @server Cluster 1600. We provide an overview of security considerations and then discuss the security terms and components in CSM for AIX.

1.1 Security overview

IBM has designed a brand new security infrastructure for the clustering environment. The approach IBM takes is to implement a security infrastructure that is easy to install and configure, but flexible enough to allow further enhancements and applications integration.

Therefore, the security infrastructure of CSM for AIX 1.3 is enabled, by default, *out of the box*. No special preparation is needed for the installation, except the standard network and infrastructure planning. There are design considerations and places to enhance security, but the node authentication is in place by default. This is in contrast to other cluster designs that take extensive configuration to achieve a secure cluster.

First, we discuss the general security considerations for several levels of the cluster environment.

In a cluster environment, it is necessary to consider the following aspects for a secure operation:

- Securing local systems (UNIX security)
- Intranet network security (firewalls and so forth)
- Securing cluster resources (RMC, RSCT, CSM)
- Securing remote command execution (for remote actions and jobs)
- General administrative tasks:
  - Graphical user interface tools (WebSM)
  - Cluster file management
  - Hardware control operations

1.1.1 System security

Just because a computer system is acting as a node in a cluster, there is no reason to ignore the standard system security procedures. See Figure 1-1 on page 3 for review of basic systems security considerations, such as:

- Non-trivial passwords
- Restrictions on who may access the system (log on)
- Restrictions on access methods (Telnet, TTY, and so on)
- Password expirations and multiple failed logon retries
- Physical security to prevent tampering with the system

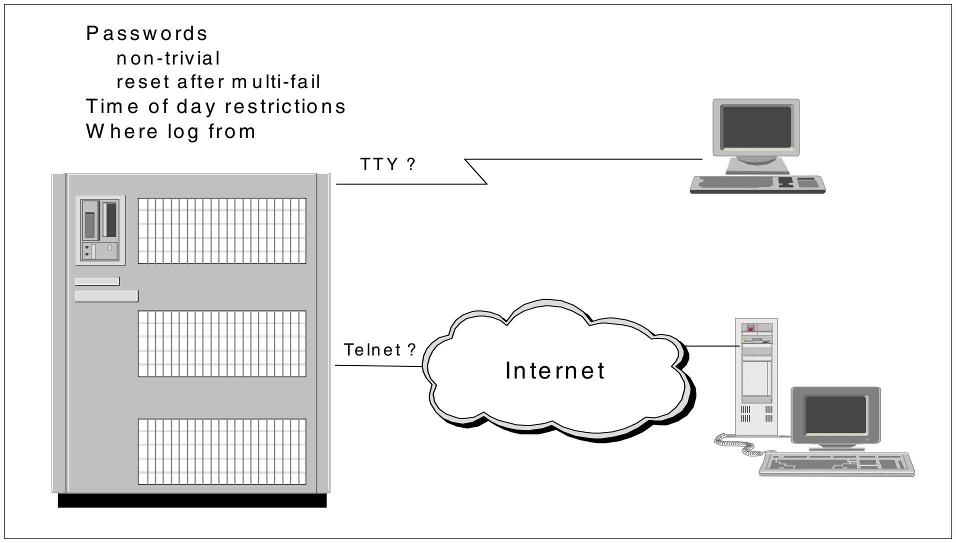


Figure 1-1 Basic system security

1.1.2 Network security basics

Basic network security rules should also be followed. There is a world of people out there on the Internet, and precautions should be taken to assure they remain “out there.” This is done by either not connecting to the Internet or by isolating the inbound network (intranet) using a firewall system connected to the outside world (Internet). See Figure 1-2 on page 4.

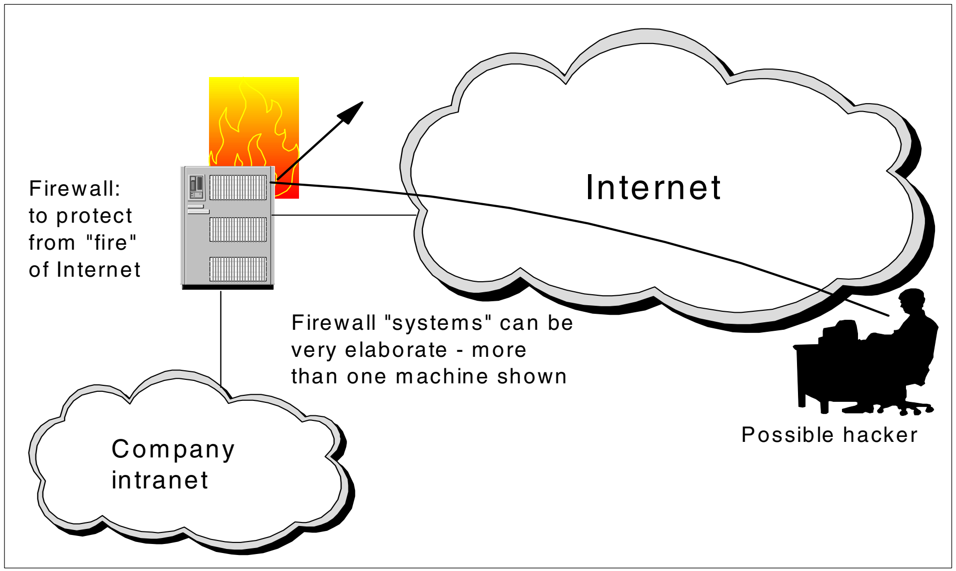


Figure 1-2 Network firewall security

1.1.3 Data transmission security

Even when the internal network and the systems are secure, there is still the concern for data transmitted over the network. You can have your data go to the right place, but there could be eavesdropping (or listening) on the wire, and the data transmitted could be copied.

To prevent eavesdropping, the data should be encrypted so that no one can understand the data even if someone is listening. See 2.3, “Security requirements and algorithm relationship” on page 18, for a more detailed description of encryption methods.

It is important for confidential information, and particularly for financial transactions, to keep the data private. Data security is especially important when you are communicating over the public Internet. See Figure 1-3 on page 5.

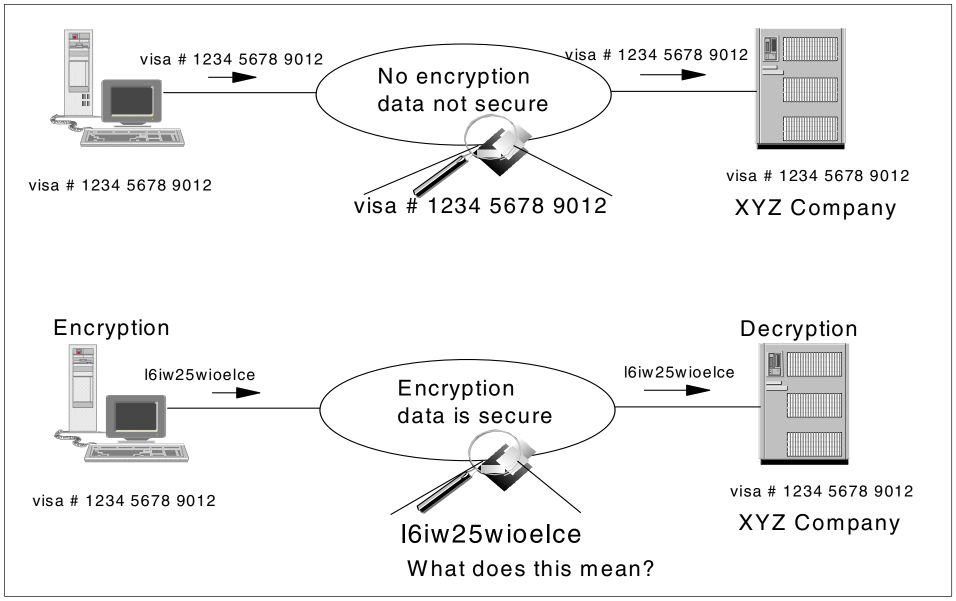


Figure 1-3 Data transmission security

In the following sections, we discuss the security components in CSM 1.3 for AIX.

1.2 Cluster Systems Management security basics

The security design of CSM is modular, with the intent of using security components that exist today, and flexible enough to incorporate new technologies and practices. The security infrastructure of CSM for AIX 1.3 is enabled, by default, out of the box.

No special preparation is needed for the installation, except the standard network and infrastructure planning. Some of the standard modules used are remote shell and distributed shell for secure commands and IBM Reliable Scalable Cluster Technology (RSCT) for secure resource authentication.

The following is a brief discussion of the main security components used in a CSM for AIX cluster.

1.2.1 Reliable Scalable Cluster Technology (RSCT)

RSCT is the “glue” that holds the nodes together in a cluster. It is a group of low-level components that allow clustering technologies, such as High Availability and General Parallel File System (GPFS) to be built easily.

RSCT technology was originally developed by IBM for RS/6000 SP systems (Scalable POWERparallel). As time passed, it became apparent that these capabilities could be used on a growing number of general computing applications, so they were moved into components closer to the operating system (OS), such as Resource Monitoring and Control (RMC), Group Services, and Topology Services.

The components were originally packaged as part of the RS/6000 SP Parallel System Support Program (PSSP) and called RSCT. RSCT is now packaged as part of AIX 5L Version 5.1 and later.

RSCT is also included in Cluster Systems Management for Linux. Now, Linux nodes (with appropriate hardware and software levels) running CSM 1.3 for Linux can be part of the management domain cluster 1600, and RSCT (with RMC) being the common interface for the clustering. For more information about this heterogeneous cluster, see *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859*.*

RSCT includes these components:

- Resource Monitoring and Control (RMC)
- Resource managers (RM)
- Cluster Security Services (CtSec)
- Group Services
- Topology Services

1.2.2 Resource Monitoring and Control (RMC)

RMC is the scalable, reliable backbone of RSCT. It runs on a single system or on each node of a cluster, providing a common abstraction for the resources of the cluster. In a cluster, the RMC framework allows a process on any node to perform an operation on one or more resources on any other node inside the cluster.

As the name implies, RMC *monitors* resources (disk space, CPU usage, processor status, application processes, and so on) and *controls* the system by performing actions in response to defined conditions.

For example, RMC can be configured to expand a file system automatically if its usage exceeds 95%. RMC enables users to configure response actions or run scripts to manage general system conditions with little or no involvement of a system administrator.

More than 80 predefined conditions are included in AIX 5L Version 5.2. They can be turned on for monitoring with a few simple commands or just a few clicks in the Web-Based System Manager (WebSM).

1.2.3 Resource managers (RM)

There is a core set of resource managers (RM) provided by RSCT, and additional resource managers are provided by CSM for AIX and CSM for Linux. Resource managers provide low-level instrumentation and control, or act as a foundation for management applications.

The following are the core resource managers of RSCT:

- Audit Log resource manager
- Configuration resource manager
- Event resource manager
- File System resource manager
- Host resource manager
- Sensor resource manager

1.2.4 Cluster Security Services (CtSec)

Cluster Security Services (CtSec) is a new security infrastructure that is used by RMC to authenticate a node within the cluster, verifying that the node is who it says it is.

This is not to be confused with authorization (granting or denying access to resources), which is handled by RMC.

CtSec uses *credential-based authentication* that enables:

- A client process to present information to the server that owns the resource to be accessed in a way that cannot be imitated.
- A server process to clearly identify the client and the validity of the information.

Credential-based authentication uses a third party that both the client and the server trust. In this release, only UNIX host-based authentication is supported.

1.2.5 Group Services and Topology Services

Group Services and Topology Services, although included in RSCT, are not used in the management domain structure of CSM. These two components are used in peer domain clusters for applications, such as High-Availability Cluster Multiprocessing (HACMP) and General Parallel File System (GPFS), providing node and process coordination and node and network failure detection. Therefore, for these applications, a .rhosts file may be needed (for example, for HACMP configuration synchronization).

These services are often referred to as *hats* and *hags*: high availability Group Services daemon (hagsd) and high availability Topology Services daemon (hatsd).

For detailed information about RSCT, refer to the RSCT product manual *IBM Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference*, SA22-7889.

2 Security concepts and components

Security in cluster environments is important to prevent unauthorized access to resources within the cluster. Before AIX 5L and RSCT 2.x, RSCT did not include a security layer. For this reason, Parallel Systems Support Program (PSSP) needs to implement Kerberos to enforce security requirements within a management domain clustering environment. CSM includes its own security layer that can be used out of the box. Therefore, there is no need to implement Kerberos.

In addition to the cluster security environment that secures the cluster communication and node membership, you still need to consider the operating system security features.

There are security features that do not need to be configured (for example, UNIX domain sockets that enable trusted and secure local host IP connections). But other features (for example, IPSec) need to be configured and are independent of the clustering environment.

This chapter describes the security requirements and mechanisms used in IBM cluster environments.

2.1 General security requirements

In the following sections, we briefly describe some general security requirements in cluster environments. These are as follows:

- Authentication
- Authorization
- Data privacy
- Data integrity

2.1.1 Authentication

Authentication is the process used to ensure that a client or server is the one it claims to be. Cluster applications, such as CSM or resource managers, need to know whether the client is really part of the cluster, or is simply attempting to obtain unauthenticated access to resources.

2.1.2 Authorization

Authorization is the process used to enable or deny user and node access to resources within the cluster. After the participating nodes have been authenticated successfully, both must ensure the access control to their resources. Basically, this process does a lookup in an access control list (ACL) to retrieve the access based on given criteria.

2.1.3 Data privacy

When transmitting data between nodes, the data should be encrypted in a way that only the acceptor can decrypt it. This is called data privacy. It is a *many-to-one* (n:1) relationship, where every node can send data to another node, but only the acceptor node can decrypt and use that data (as shown in Figure 2-1 on page 11). If the data has been intercepted by another node, that node will not be able to decrypt it.

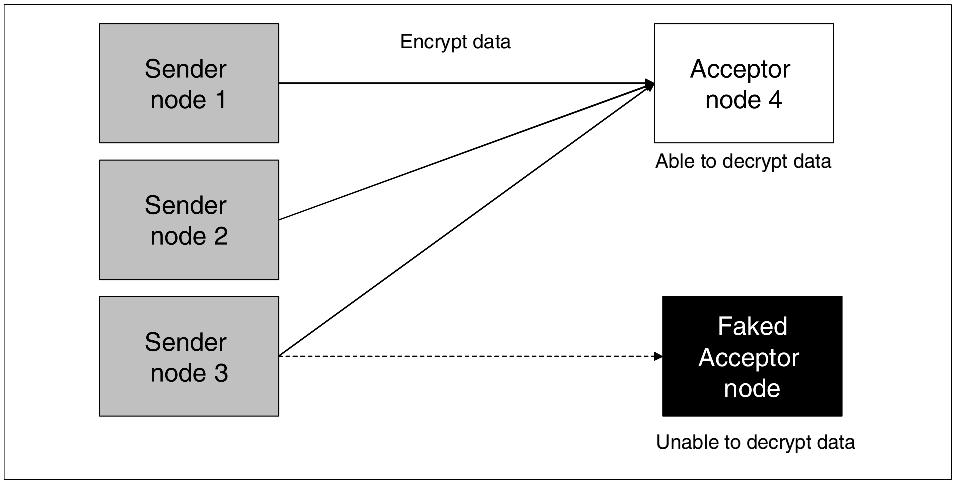


Figure 2-1 Data privacy example

2.1.4 Data integrity

To ensure that received data is from the expected sender node, data must be encrypted in a way that the acceptor can be sure that the received data can only come from the expected node. This is called data integrity. It is *one-to-many* (1:n) relationship, where the sender encrypts the data and everybody receiving this data can decrypt it (as shown in Figure 2-2 on page 12). Because any node that has intercepted the data can decrypt it, this data should not contain sensitive information.

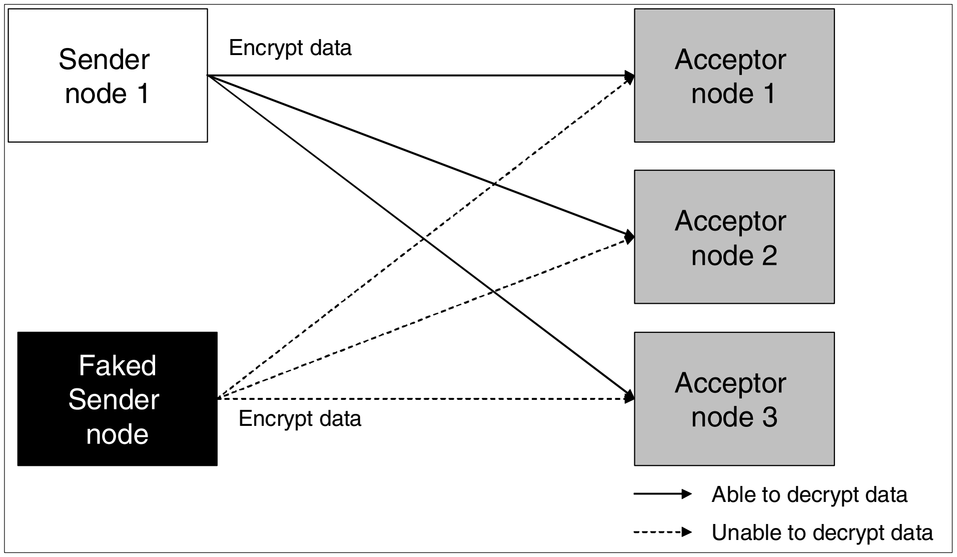


Figure 2-2 Data integrity example

2.2 Security algorithms

There is a certain level of security that is associated with each node at the operating system level that is basic and almost assumed, but should not be forgotten.

- User ID: Be sure to set up passwords and only give the authorization level the users need.
- Firewall security from the public Internet.

When sending information between computers over a network, there are additional security concerns:

- Who can see the information?
- Who can steal the information?
- Am I communicating with the right party?
- Is the received data unaltered?

In order to provide secure data transmission, several encryption methods or algorithms have been developed. The following sections present a brief overview of some of these techniques. Sometimes, more than one encryption method is used for more complete solutions.

2.2.1 Symmetric key encryption

Symmetric key encryption is basically when both ends of a communication channel have the same key (they are symmetric). This way, after both ends agree on the key, and they both have the key, data exchange is secure as long and they do not give their key away. The benefits of symmetric key encryption are that it is fast and has low system overhead (takes less computing power). There is a problem, though: how to securely share the keys. See Figure 2-3.

Some standard encryptions that follow this method include:

- Data Encryption Standard (DES): 56-bit encryption, most common.
- Commercial Data Masking Facility (CDMF): Designed by IBM (similar DES, but uses a 40-bit key).
- Triple DES (3DES): Similar to DES, but uses a triple encryption using different keys.
- International Data Encryption Algorithm (IDEA): Similar to DES, but it has a larger, 128-bit, key.
- RC2/RC4: By Ron Rivest, RSA Data Security.

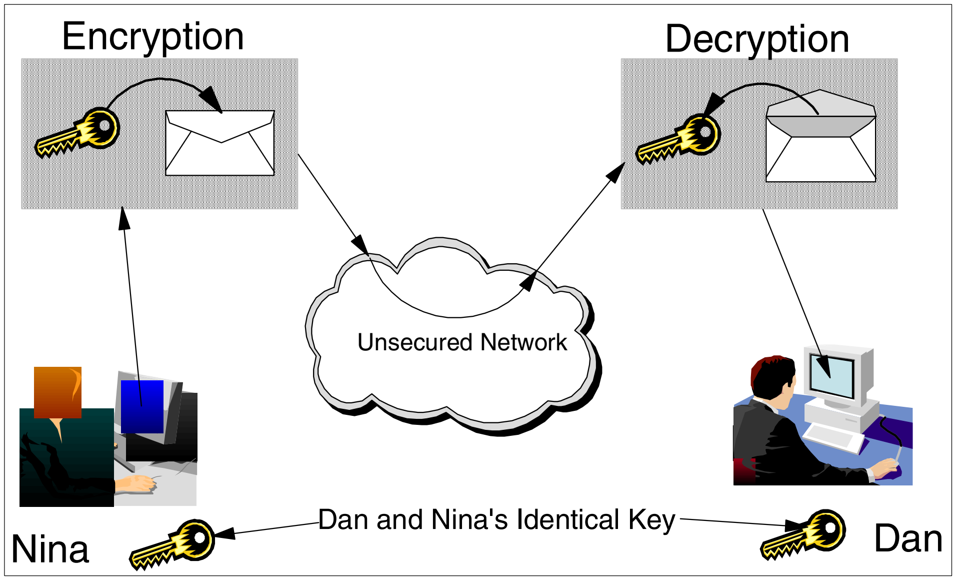


Figure 2-3 Symmetric key encryption

2.2.2 Public key encryption

Public key encryption, also known as public-private key, uses a multistep process to transmit encrypted data:

1. The receiver of the information gives his or her *public* key to the sender.
2. The sender encrypts the data using the receiver's public key.
3. The receiver gets the encrypted data and decrypts it using his or her private key.

This method eliminates the problem of sharing the key publicly, but takes a penalty in system overhead. This is a widely used encryption method, mainly using the Rivest, Shamir, and Adleman (RSA) algorithm by RSA Data Security, Inc.

Figure 2-4 illustrates the public key method. For large data transfers, this method is often used to share a symmetric key that is then used for high speed and lower overhead symmetric key encryption.

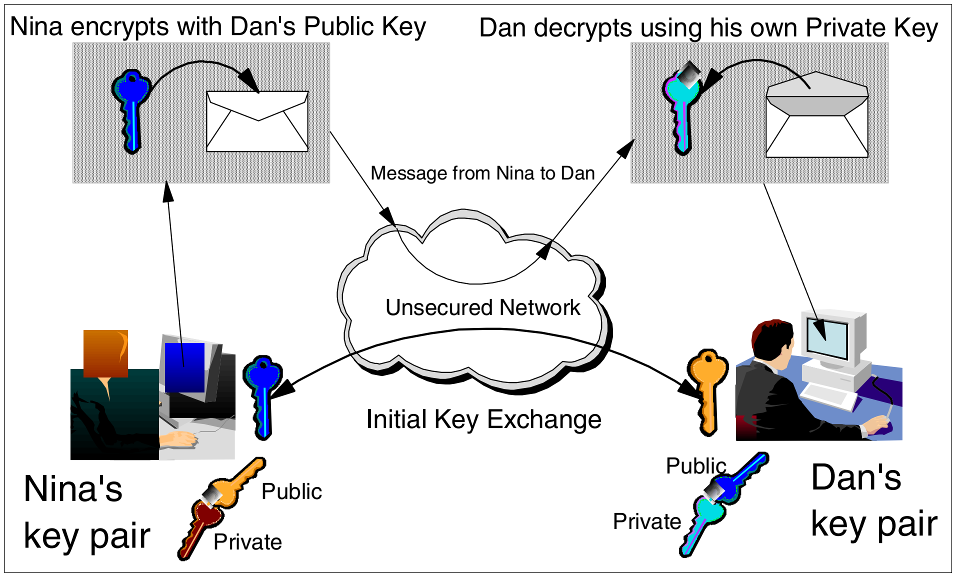


Figure 2-4 Public key encryption

2.2.3 Secure hash functions

The *hash function* is also know as a one-way hash, message digest (MD), fingerprint, or compression function. It uses an algorithm to compress a variable-length message into a fixed-length binary.

This is used especially for digital signatures, but can also be used for tasks, such as securely storing passwords in a database. One-way function means that there is *no* way to recompose the original message from its digest.

Examples of hash function encryption include:

- MD4/MD5: 128-bit, Message Digest by Ron Rivest
- SHA: 160-bit, Secure Hash Algorithm designed by National Institute of Standards and Technology (NIST) and National Security Administration (NSA)

Figure 2-5 illustrates a diagram of the hash function.

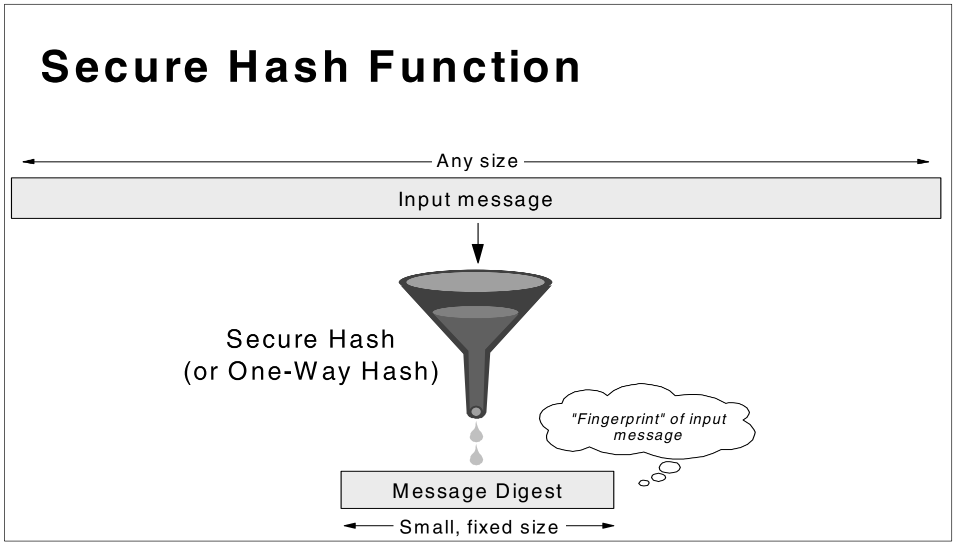


Figure 2-5 Secure hash function encryption

Creating digital signatures with the hash function assures the integrity of the data being transmitted. How this works is illustrated in Figure 2-6 on page 16.

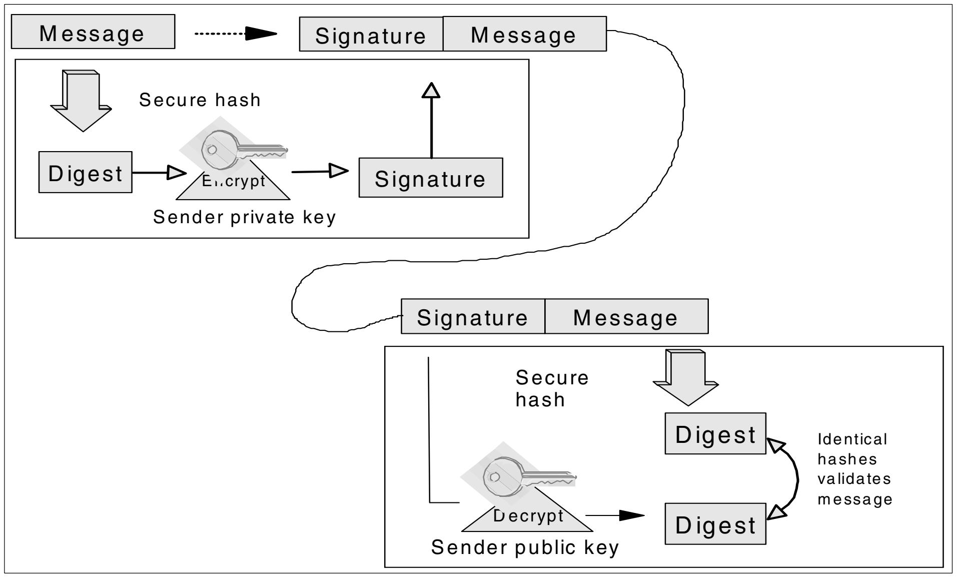


Figure 2-6 Digital signature

2.2.4 Public key certificate

The use of public key certificates comes from the need for an a independent third party for the actual certification of a user's public key when that public key is passed to another party.

This certifier creates a certificate that contains the public key of a user and attaches its digital signature. When this certificate is received, along with the corresponding public key, the receiver can be sure that the key is actually from the expected sender.

The certifier is, in fact, a trusted host. See Figure 2-7 on page 17 for diagram of certificates.

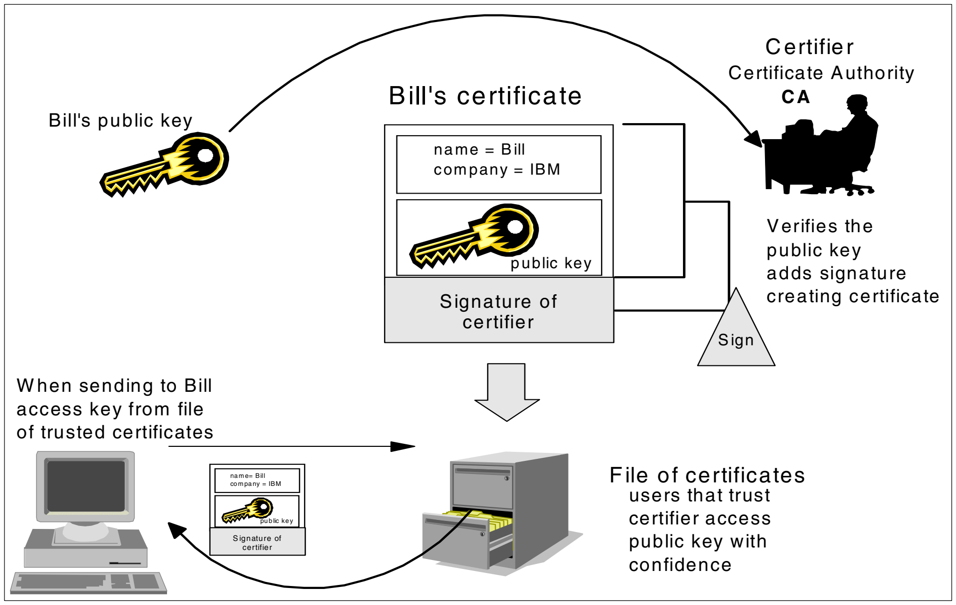


Figure 2-7 Public key certificate

2.2.5 Secure Sockets Layer and Transport Layer Security

Secure Sockets Layer (SSL), and its successor Transport Layer Security (TLS), encryption is a combination of several encryption methods used in a Web server and Web client environment. SSL and TLS use public-private keys and digital certificates (a hash function). They operate within the TCP/IP stack between the Transmission Control Protocol (TCP) and the Hypertext Transfer Protocol (HTTP).

Secure Shell (SSH) also uses SSL.

Examples of SSL and TLS include most of the Internet server and browser packages, such as Netscape, Microsoft Internet Explorer and so on.

See Figure 2-8 on page 18 for a diagram of the internals of SSL.

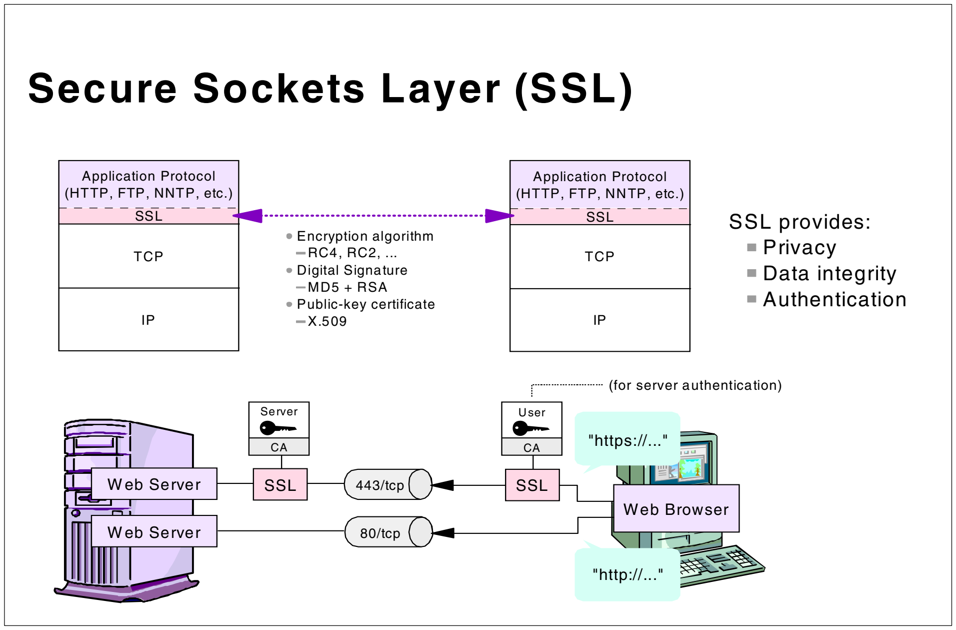


Figure 2-8 Secure Sockets Layer example

2.2.6 Secure Shell (SSH)

Secure Shell (SSH) operates in a UNIX environment similar to the remote shell **rsh** command, but with encrypted transmissions. Unlike rsh, which depends on the .rhosts file, **ssh** uses the SSL encryption and bypasses the .rhosts file entirely.

OpenSSH is a free version of SSH developed by the OpenBSH project.

2.3 Security requirements and algorithm relationship

The following sections describe the security requirements and algorithm relationship.

2.3.1 Using encryption to ensure data privacy

To ensure data privacy, data must be encrypted in the way that only the acceptor can decrypt it. Basically, data is encrypted with the public and private key method.

If a node wants to send data to another node, it first requests the public key of the node that the data is intended for. Then, the data is encrypted with this public key. Now, decrypting is possible only with the private key of the acceptor node. This node retrieves the encrypted data and is able to decrypt and accept it with its own private key.

2.3.2 Using signatures to ensure data integrity

To ensure that retrieved data is from the expected node, data must be encrypted in a way that indicates the received data is from the sender node. This is called data integrity, and the data is signed with the public and private key method.

If a node wants to sign data, it uses its own private key. Usually, this method creates a checksum or one-way hash of the data and encrypts it. This encrypted hash or checksum is called a *signature*. The acceptor of the signed data now is able to decrypt and accept the signature with the sender's public key.

Because data encrypted with a private key can be decrypted by everyone that has access to the sender's public key, it should only be used to ensure that the node is the one it claims to be and should not contain sensitive data.

2.3.3 Combining data integrity and data privacy

The most secure way of transmitting data is using data integrity and data privacy. This enables the acceptor to trust that the data comes from the node it expects and it allows the sender to trust that the data can be read only by the acceptor.

How this method affects data blocks is illustrated in Figure 2-9 on page 20.

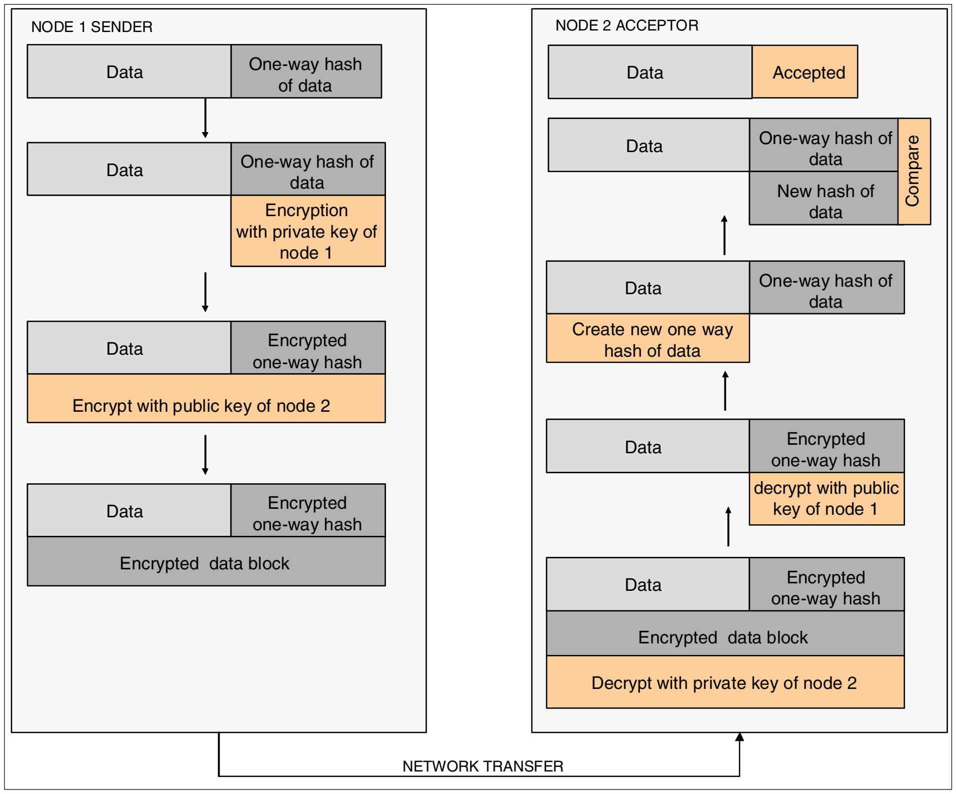


Figure 2-9 Data transfer using data integrity and data privacy

To apply data integrity by signing the data:

1. Create a one-way hash of the data.
2. Encrypt the one-way hash with its private key.

To apply data privacy by encrypting data:

1. Encrypting data and the signature with the acceptor's public key.

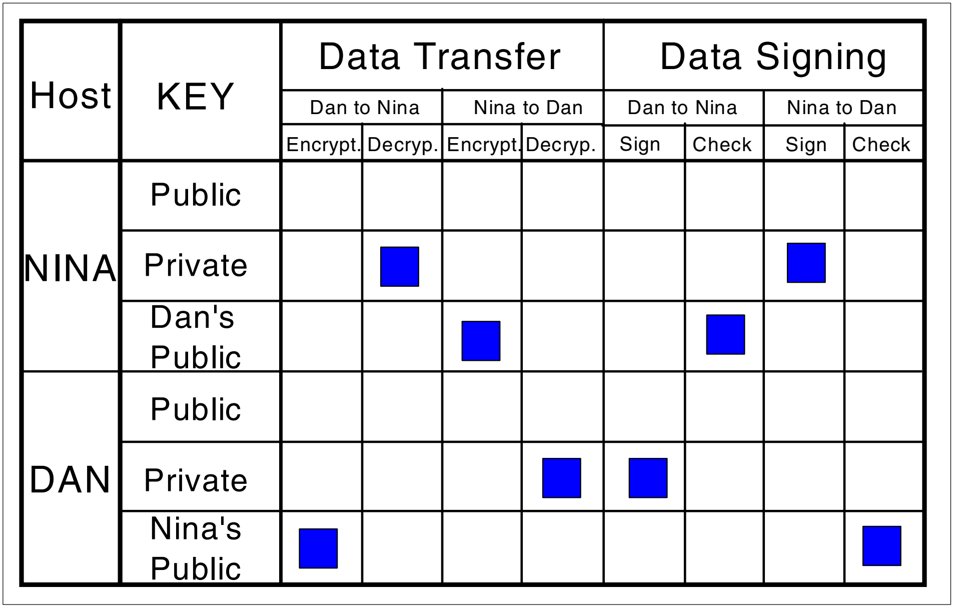
To send data to the acceptor, the acceptor completes the following steps:

1. Decrypts the data block by using its private key.
2. Decrypts the one-way hash by using the sender's public key.
3. Generates a new one-way hash of the data.
4. Compares the hash in the data block with the newly generated hash.

If both hashes are same, the data transfer occurs without failures and is secure, and the data will be accepted.

2.3.4 Use of different cryptographic techniques

Figure 2-10 shows the relationship between security methods and algorithms in order to achieve both data integrity and data privacy.



Host	KEY	Data Transfer				Data Signing			
		Dan to Nina		Nina to Dan		Dan to Nina		Nina to Dan	
		Encrypt.	Decryp.	Encrypt.	Decryp.	Sign	Check	Sign	Check
NINA	Public								
	Private		🟦					🟦	
	Dan's Public			🟦			🟦		
DAN	Public								
	Private				🟦	🟦			
	Nina's Public	🟦							🟦

Figure 2-10 Secure data transfer between two parties

3

Cluster Systems Management security infrastructure

In this chapter, we describe the terminology and concepts used in CSM and RSCT security. We also describe the RSCT security architecture and mechanisms used by CSM. Knowledge of RSCT and its base components (such as RMC, Resource managers, and so forth) is assumed.

It is important to understand that most of the components and mechanisms explained in this chapter do not need to be configured. CSM and the security layer work out of the box. This chapter explains what happens underneath the CSM commands.

3.1 Reliable Scalable Cluster Technology security

CSM clusters are management domains clusters configured on top of RSCT.

Note: In this section, we consider the requester node (the node that requests access to a resource within the cluster) as the *client*, and the node that owns the resource (and serves it) as the *server*.

The CSM management server contacts its managed nodes using RSCT components to ensure they are still alive or to request access to their resources. Every time an application or component, for example, an RSCT resource manager (RM), of the management server sends a functional request to a resource, either local or remote, the RSCT subsystem is called.

In the case of CSM, the RMC functions are used, and the requests are sent to the node where the resource is located. See Figure 3-1.

Because remote connections are simply TCP/IP socket connections, they must be secured in order to ensure that both the requestor and the server node is the one the cluster expects it to be. This is where the Cluster Security Services (CtSec) comes into play.

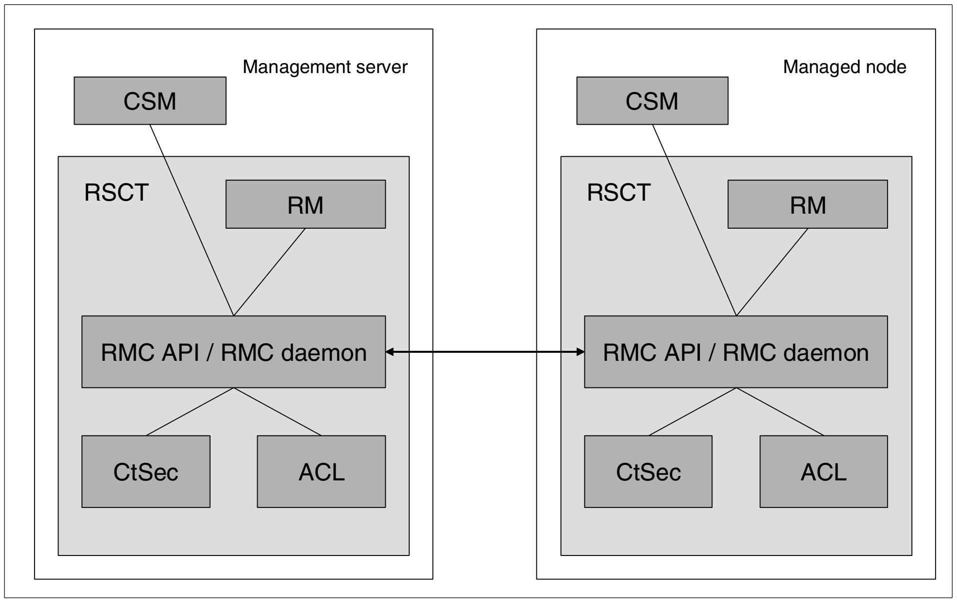


Figure 3-1 Basic CSM cluster communication overview

The communication between cluster members is a client/server communication. In the following sections, *client* is used for an application, for example, a local resource manager or a CSM command that is using the RMC client API.

These client applications are linked against the RMC and CtSec shared libraries. If those clients request access to resources on a remote node, the RMC daemon on that remote node will be the *server* for those requests.

Cluster Security Services (CtSec) overview

Cluster Security Services (CtSec) are integrated into the RSCT subsystem and are used by RMC to determine the identity of a client from a node. This authentication process results in a security context that is used by RMC for the communication between the participating nodes to fulfill the client's request.

Note: The security context is at a client/server level, not at a node level.

To create this security context, CtSec uses credentials for the authentication. Those credentials are used to determine the authenticity of a node and the client application. To access resources on a remote node, both nodes send and compare the credentials during the authentication process.

This process allows the following:

- A client to present information about itself that cannot be imitated by others
- A server to clearly identify the client by its given credentials
- A client to be sure it is retrieving data from the correct server

The credential-based authentication involves other components to create and identify credentials. This component-based architecture, shown in Figure 3-2 on page 26, also allows future extensions to the security layer in RSCT.

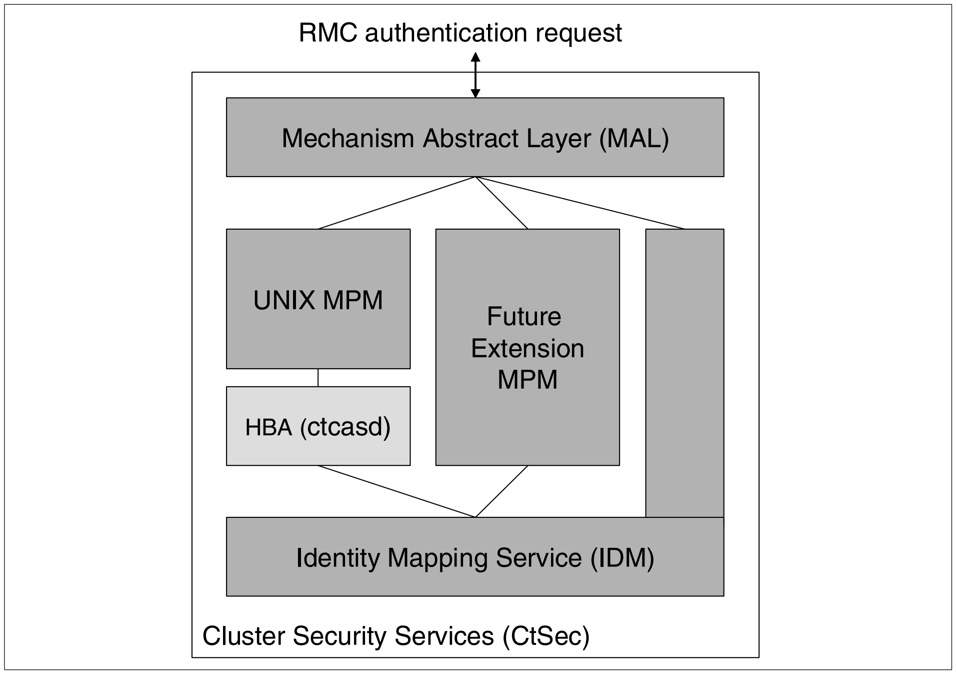

RMC authentication request
Mechanism Abstract Layer (MAL)
UNIX MPM
HBA (ctcasd)
Future Extension MPM
Identity Mapping Service (IDM)
Cluster Security Services (CtSec)

Figure 3-2 Cluster Security Services (CtSec) architecture

Within RMC, CtSec is responsible for authentication only. RMC itself is responsible for authorization by using an access control list (ACL) to grant or deny access to resources within the cluster.

3.2 Components of Cluster Security Services (CtSec)

The CtSec library consists of several components required to provide the current and future security functions.

The security context, created by CtSec, contains credentials for both the client and the server, the state of authentication (authenticated or unauthenticated), and the session information (for example, session key or expiration time). The security context is created by the components of CtSec and is sent to RMC as a result of the CtSec authentication.

3.2.1 Mechanism abstract layer (MAL)

CtSec exports an interface to applications that need to implement cluster security. This interface is called mechanism abstract layer (MAL). MAL provides a generic, mechanism-independent interface for the underlying security mechanisms. MAL sends these general instructions to the configured security modules.

Those pluggable security mechanism modules are called mechanism pluggable module (MPM). The result of MAL and the configured MPM is the security context that contains credentials and possibly a session key used by both parties involved in the communication process.

If a client turns off authentication by setting the environment variable CTSEC_CC_MECH=none, MAL does not involve MPM for authentication, but returns an unauthenticated security context as the result of the authentication process.

The configuration of MPMs is done in the /var/ct/cfg/ctcec.cfg file. This file contains all MPMs CtSec should use during the authentication process. In future releases, IBM will consider the development of other security modules. Module usage is designed to be based on predefined priorities.

If a client requests access to resources, MAL is used to initiate the security context. If this process is successful, this context is cached by the MAL layer for later requests, and it is retained in memory until it is terminated by the application.

3.2.2 Mechanism pluggable module (MPM)

Each mechanism pluggable module (MPM) converts the general tasks, received from the MAL layer, into necessary tasks the security mechanism uses to satisfy the MAL request.

The MPM gathers all credentials that are necessary to fulfill the authentication process for this specific security mechanism. The MPM also maps network identities to local identities (see 3.2.5, “Identity mapping service” on page 29).

Within CSM 1.3, only one MPM is supported. This MPM is called UNIX MPM. UNIX MPM has been developed for both 32- and 64-bit operation, depending on the OS kernel. Due to the modular architecture of MAL and MPM, other security mechanisms may be added in future releases. MPMs are object modules that are loaded by the MAL during run time.

MPMs are located in the /usr/sbin/rsct/lib directory. Each MPM must have a link in /usr/lib/ that points to the respective file in /usr/sbin/rsct/lib/ (see Example 3-1 on page 28).

Example 3-1 Location of MPMs

```
ls -al /usr/sbin/rsct/lib/*.mpm*
-r--r--r--   1 bin   bin   192120 Sep 06 13:26 /usr/sbin/rsct/lib/unix.mpm
-r--r--r--   1 bin   bin   199952 Sep 24 11:27 /usr/sbin/rsct/lib/unix.mpm64
ls -al /usr/lib/*.mpm*
lrwxrwxrwx 1  root  system  27 Oct 11 10:22 /usr/lib/unix.mpm ->
                                            /usr/sbin/rsct/lib/unix.mpm
lrwxrwxrwx 1  root  system  29 Oct 11 10:22 /usr/lib/unix.mpm64 ->
                                            /usr/sbin/rsct/lib/unix.mpm64
```

3.2.3 UNIX mechanism pluggable module

Actually, the UNIX MPM (which uses host-based authentication) is the only MPM implemented in CtSec and used by RSCT. The core of the UNIX MPM is the ctcasd client API (see 3.2.4, “Host-based authentication with ctcasd” on page 28), which creates the host-based authentication (HBA) credentials.

The ctcasd component is called only if TCP/IP sockets for remote connections are used. If the request is for the local host, UNIX MPM uses UNIX domain sockets within the kernel. The kernel security is trusted, and no further security features are required.

3.2.4 Host-based authentication with ctcasd

The Cluster Technology Cluster Authentication Service daemon (ctcasd) creates credentials based on the node’s host name and client identity. Remember that a *client* is an application that uses the RMC client API.

Additionally, ctcasd creates a session key that can be used as a symmetric key for the node communication after finishing the authentication. For this release, the session key is created, but not used. This session key is created to allow further applications to encrypt and decrypt data using a symmetric key algorithm, which is faster than public and private key (PPK) algorithms.

To ensure data privacy for the secret session key and data integrity for the host credentials, ctcasd uses the host’s public/private key pair to encrypt and decrypt this information. In actual implementation, this key pair is generated with the RSA512 algorithm. It is possible to change this to a 1024-bit key by editing the ctcasd configuration file (/var/ct/ctcasd.cfg).

To ensure that ctcasd uses the correct public key during the encryption process, the public keys in the trusted host list (THL) file are associated with the host name of the node. For this reason, it is necessary that all nodes within the CSM cluster (including the management server) resolve names identically.

Important: To ensure identical host name resolution, all participating cluster members should use a method for name resolution that gives *identical* results on all nodes in a CSM cluster. The name resolution method and order can be changed in /etc/netsvc.conf (for AIX systems). All hosts should also use either short or fully qualified host names. If the cluster consists of nodes in different domains, fully qualified host names *must* be used.

First, ctcasd encrypts the session key with the target host’s public key derived from the THL file. This ensures that only the target node can decrypt this session key with its private key, and data privacy is ensured.

To ensure data integrity for the host credentials, ctcasd now encrypts the whole credential data structure using the initiator’s private key. Everyone can decrypt the data block with the sender’s public key, but the target node can be sure it was sent by the expected node.

The distribution of public keys to all nodes is performed by CSM. By adding a node to the CSM cluster, CSM runs RSCT commands to achieve the public key exchange between the management server and its nodes.

The public key exchange is done over the network. During this exchange, the network must be secure against tracing and spoofing, because the keys are binded to a node within the cluster.

Important: With CSM, it is not possible to disable the public key exchange. If you feel that your network is not secure enough, you should verify the keys on the management server and nodes manually. To verify the keys, refer to 6.1.8, “Verifying exchanged public host keys” on page 89.

In the current version, the host keys generated by ctcasd do not expire, but they can be updated manually as described in 6.1.4, “The ctcasd daemon key files” on page 84.

3.2.5 Identity mapping service

The identity mapping (IDM) service result is used for authorization. The IDM service maps the network identities to local identities if there is a mapping rule specified in the configuration files (called maps).

This local identity can be used by RMC to retrieve the permission from the RMC access control list (ACL). To see how RMC gathers permissions for resources from the ACL, see 3.2.6, “Resource Monitoring and Control access control list” on page 31.

Because, by default, there is no common user space inside a CSM cluster, same user names on different hosts might not have the same permissions to access resources. It is also possible that one user spans across the whole cluster.

As described in 3.2.6, “Resource Monitoring and Control access control list” on page 31, RMC gathers permissions for resources by using the client’s network identity first.

For example, if user *david* should exist on all nodes within a 128-nodes cluster and should have access to a resource on a specific node, the RMC ACL file on that node must contain an entry for each network identity of the client. Example 3-2 shows the output for a specific resource in the RMC ACL file for 128 nodes.

Example 3-2 Output of /var/ct/cfg/ctrmc.acl for a specific resource entry

```
IBM.FileSystem
  david@node001   *   rw   # access for david from node 001
  david@node002   *   rw   # access for david from node 002
  david@node003   *   rw   # access for david from node 003
  david@node004   *   rw   # access for david from node 004
  .
  .
  .
  david@node128   *   rw   # access for david from node 128
```

This is the main reason for which IDM was designed. IDM simply maps network identities in a management cluster to a local identity in order to avoid hundreds of lines in the RMC ACL file.

CSM configures a mapping configuration file, /var/ct/cfg/ctsec_map.global, that contains the mapping relationship between local and network identities.

To follow our example, the mapping file will need only one line to map the user david on all the cluster nodes to a single local identity, called mapped_david:

```
unix:david@<cluster>=mapped_david
```

This maps the user david, coming from every node in the current active cluster, to the local identity mapped_david for mappings initiated by the UNIX MPM. This local identity does not need to exist as a real user in the operating system.

Access to resources works even if this identity does not exist locally (in /etc/passwd).

With this mapped identity, the resource in the ACL file needs only one entry, as shown in Example 3-3 on page 31.

Example 3-3 RMC ACL file using local identities

```
IBM.FileSystem
  mapped_david   *   rw   # access for david from node 001
```

Within the ACL file, you can easily distinguish between network identities and local mapped identities. Network identities always contain an @ and the host name or node ID where the client comes from. Local mapped identities only consist of a specifier, because the host name has already been specified in the mapping file.

Initially, the global mapping file contains these entries, as shown in Example 3-4, (for example, unix: specifies the MPM used for authentication, and =root specifies the local identity).

Example 3-4 Global mapping file

```
cat /var/ct/cfg/ctsec_map.global
unix:root@<cluster>=root
unix:root@<any_cluster>=any_root
unix:*@LOCALHOST=*
```

Basically, the first line says that every request to RMC, coming from root at any node within the current active cluster is mapped to the local identity root. Every root request from nodes outside the active cluster is mapped to any_root.

Every client request from local host to local host resources, from any user, is simply mapped to the local user. The mapping file lookup process is based on a first-matching basis, that means the first identifier that fits the given network identity will be mapped to the appropriate local identity, and no further mapping for the same network identity will occur.

3.2.6 Resource Monitoring and Control access control list

RMC uses an access control list (ACL) to grant or deny access to resources within the CSM cluster. The ACL is updated by CSM during the cluster administration procedures (for example, adding and removing nodes).

After RMC successfully authenticates the client, it makes a lookup into the ACL to get the permissions on the given criteria (see Figure 3-1 on page 24).

The original RMC ACL file is located in /usr/sbin/rsct/cfg/ctrmc.acl. CSM does not use this original file, instead it copies it to /var/ct/cfg.

For normal cluster operation, RMC look up in /var/ct/cfg/ctrmc.acls first and uses this ACL file if it exists. Otherwise, it uses the original file in the RSCT directory.

By default, you do not need to copy and change the ACL file manually. CSM does it while maintaining the cluster. After the installation of the management server, an ACL exists in /var/ct/cfg, and it is used by RMC.

The ACL file consists of stanzas containing the resource classes and the defined permissions of users and hosts within the cluster. RMC uses two different identities for the retrieving the permissions stored in the ACL. These identities are as follows:

- The client's network identity presented by the client's user name and its host name, for example, root@node1
- The local mapped identity, as described in the 3.2.5, "Identity mapping service" on page 29

The order of the lookup process in the ACL file is first the network identity and then local identity.

3.3 Communication flow examples

The following sections explain the communication flow for several scenarios for a CSM cluster setup.

3.3.1 Initial cluster setup

During the management server installation process the **`installms`** and **`csmconfig`** commands are issued. These commands install the CSM management server filesets and configure basic cluster properties. From an RSCT point of view, this node still is an independent workstation. The node does not become a management server until it has at least one node to manage.

3.3.2 Adding a new node

The CSM management server takes care of the RSCT security environment while adding nodes to the cluster. This action is transparent to the user. You do not have anything to configure, because everything is performed by CSM automatically. When adding a new node to a CSM cluster, the **`definenode`** and **`updatenode`** commands are issued.

The following steps are done by **`definenode`**:

1. The CSM management server adds a node definition to the CSM management cluster and database by using its RMC client API.
2. Within the node definition, the attribute AllowManageRequest for this node definition is set to 1.

The following steps are done by **`updatenode`**:

1. Through a remote shell command issued to the managed node, CSM creates the management server resource definition (IBM.ManagementServer) and defines the node as a managed node by using its RMC client API. The node RMC calls the management server RMC and sends a request to be managed. RMC exchanges credentials, for example, connectivity node name, host name, public key, node identifier, and so on.
2. The management server uses an unauthenticated request to the node in order to retrieve the public key of that node, and then it stores the public key and the host name in the trusted host list (THL) using its ctcasd API.
3. The management server sends the management node host name, management node identifier, and the management node's public key to the managed node.
4. The node stores the public key and host name of the management server in its THL file.
5. The managed node RMC updates its ACL files to allow the root@nodeid and root@management_server_hostname identities read and write access to all resources. All other users get read-only access.

3.3.3 Requesting access to resources

The following steps show the communication flow between the management server and a node requesting access to a resource. Figure 3-3 shows a simple illustration of the given example.

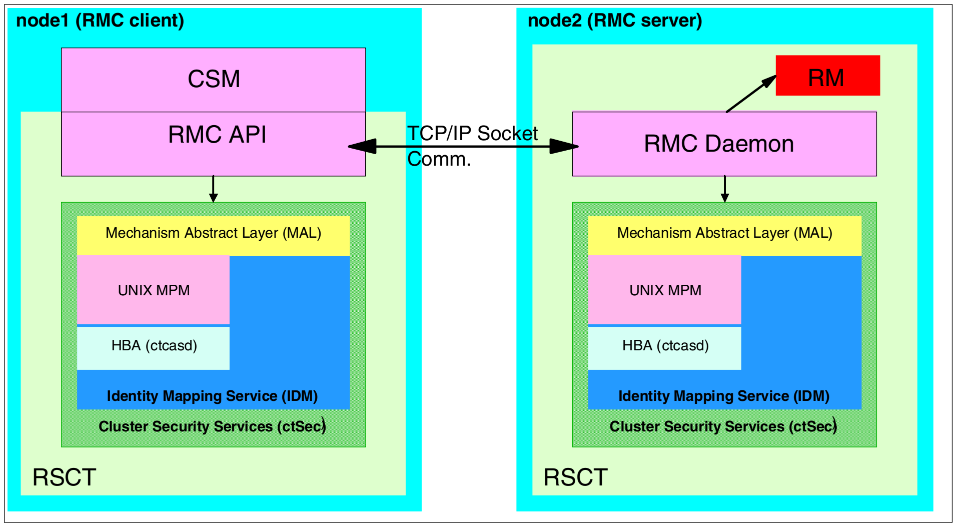

node1 (RMC client)
CSM
RMC API
Mechanism Abstract Layer (MAL)
UNIX MPM
HBA (ctcasd)
Identity Mapping Service (IDM)
Cluster Security Services (ctSec)
RSCT
TCP/IP Socket Comm.
node2 (RMC server)
RM
RMC Daemon
Mechanism Abstract Layer (MAL)
UNIX MPM
HBA (ctcasd)
Identity Mapping Service (IDM)
Cluster Security Services (ctSec)
RSCT

Figure 3-3 Example of CSM security communication

The communication flow between the management server and a node requesting access to a resource is as follows:

1. CSM on node1 wants to access a resource on node2. Because CSM uses RMC client APIs, CSM is a client that has specific process properties. In this case, the RMC daemon on node2 acts as a server that accepts or rejects the request from the client.
2. The client on node1 needs to create a security context to be able to communicate with RMC on node2. This is done by calling an RMC API function.
3. RMC on node1 calls CtSec to initiate a security context. CtSec inherits properties of the client that initiates the security context (for example, the user of the calling process).
4. RMC on node1 calls the MAL component to initiate the security context.
5. MAL on node1 calls the MPM to initiate the security context.
6. UNIX MPM on node1 initiates the generation of credentials for the client by using HBA/ctcasd.
7. HBA/ctcasd on node1 also generates a session key that can be used for further data transfers.

8. Then, ctcasd on node1 looks up in its THL file for the public key of node2 (target host). This public key is used to encrypt the session key, which must be kept secret for both parties to ensure further data privacy, as described in 2.1.3, "Data privacy" on page 10. This encrypted session key can only be decrypted by node2. If the public key for node2 is not found, the credentials will not contain a session key.

   Next, ctcasd on node1 encrypts the credentials with its private key. This is done for data integrity, as described in 2.1.4, "Data integrity" on page 11. Figure 3-4 shows the credentials and how they are encrypted.

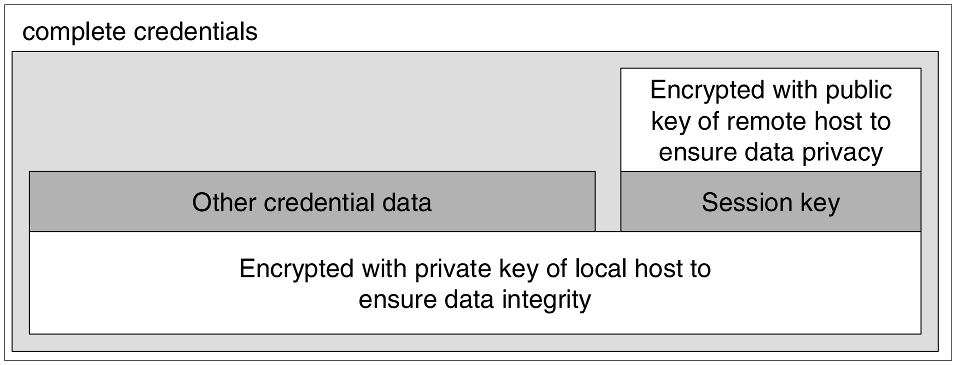

complete credentials
Encrypted with public key of remote host to ensure data privacy
Other credential data
Session key
Encrypted with private key of local host to ensure data integrity

Figure 3-4 The ctcasd credentials after encryption

9. HBA/ctcasd on node1 return credentials to MPM on node1.
10. MPM wraps credentials into a Control Context Data Buffer (CCDB) and returns it to MAL on node1.
11. MAL on node1 returns the CCDB to RMC on node1.
12. RMC on node1 sends the CCDB to RMC on node2 to establish the security context.
13. RMC on node2 passes the received CCDB to MAL on node2.
14. MAL on node2 determines the involved MPM based on the MPM code within the CCDB, and passes it to the appropriate MPM.
15. UNIX MPM on node 2 involves ctcasd to authenticate credentials.
16. Then, ctcasd decrypts the credentials with the public key of node1. If the CCDB contains a session key, ctcasd will decrypt the session key with its private key. Finally, ctcasd returns the CCDB to the same MPM.
17. MPM calls IDM for resolving the local identity and network identity of the client, updates the CCDB, and returns it to MAL.
18. MAL builds a security context out of the CCDB and returns it to RMC on node2.

19. RMC on node2 sends the server credentials for authentication to node1 in the same way, because it is a mutual authentication.
20. After the authentication of the server, both parties update their security context that the authentication was successful.
21. RMC on node2 retrieves the permissions from its ACL file to grant or deny access to the resource requested by node1.
22. If granted, RMC on node2 calls the appropriate resource manager to fulfill the request.
23. The data is then sent back to the client on node1.
24. The security context is closed on both nodes.

4

Practical security considerations

This chapter describes some of the practical security considerations that a systems administrator should understand before building a CSM cluster.

We discuss the following topics:

- Network considerations
- Shell security (required parameters)
- Configuration file manager (CFM)
- User management
- Security in a heterogeneous environment
- Web-Based System Manager
- Security considerations for hardware control
- Name resolution

4.1 Network considerations

For both performance and security reasons, it is important to understand and control the data that is transferred around your network. If the data you transmit over a network is sensitive, and is not encrypted, consider isolating that data on a dedicated network.

It is good practice to isolate the management server, Hardware Management Consoles (HMCs), console servers, and RSAs on a dedicated network. In a CSM environment, this network is known as a *management virtual LAN* (VLAN). It hides the HMCs, console servers, and RSAs from the normal users and keeps the traffic between them from being visible on the public network.

The CSM management servers uses Telnet, which transmits passwords in clear text, to communicate with console servers.

Many systems administrators choose to create a network for cluster administration data, such as node installations, backups, and monitoring. In a CSM environment, this network is known as a *cluster VLAN*.

Many systems administrators also create a network for application-related traffic. In a CSM environment, this network is known as a *public VLAN*.

It is possible to use private IP addresses (192.168.*.*, 172.16.*.*, or 10.*.*.*) for both the management VLAN and the cluster VLAN. Private addresses can provide a higher level of security than public addresses. However, if your administrators' workstations are not on the management VLAN, they will have no direct access to HMCs, RSAs, or console servers. They will have to access them by logging on to the management server. If this is not acceptable, the only alternative is to use routing. This routing can become complex if your administrators' workstations are dispersed across multiple networks, or if they use DHCP for their workstations.

Incorrectly implemented routing can impact security, and thus defeat the purpose of creating multiple networks. For example, incorrect routing can cause secure data to be transmitted over the public VLAN.

Figure 4-1 on page 39 shows a possible network configuration for a fairly simple environment. It contains three networks, a management VLAN, a cluster VLAN and a public VLAN:

- The management VLAN contains only the CSM management server, HMCs, console servers, and RSAs.
- The cluster VLAN contains the management server and the nodes.

- The public network contains the nodes, the users' workstations, and the administrators' workstations. It may also contain other servers that are not members of the CSM cluster.
- The HMCs, console servers, and RSAs can be accessed only through the management server.

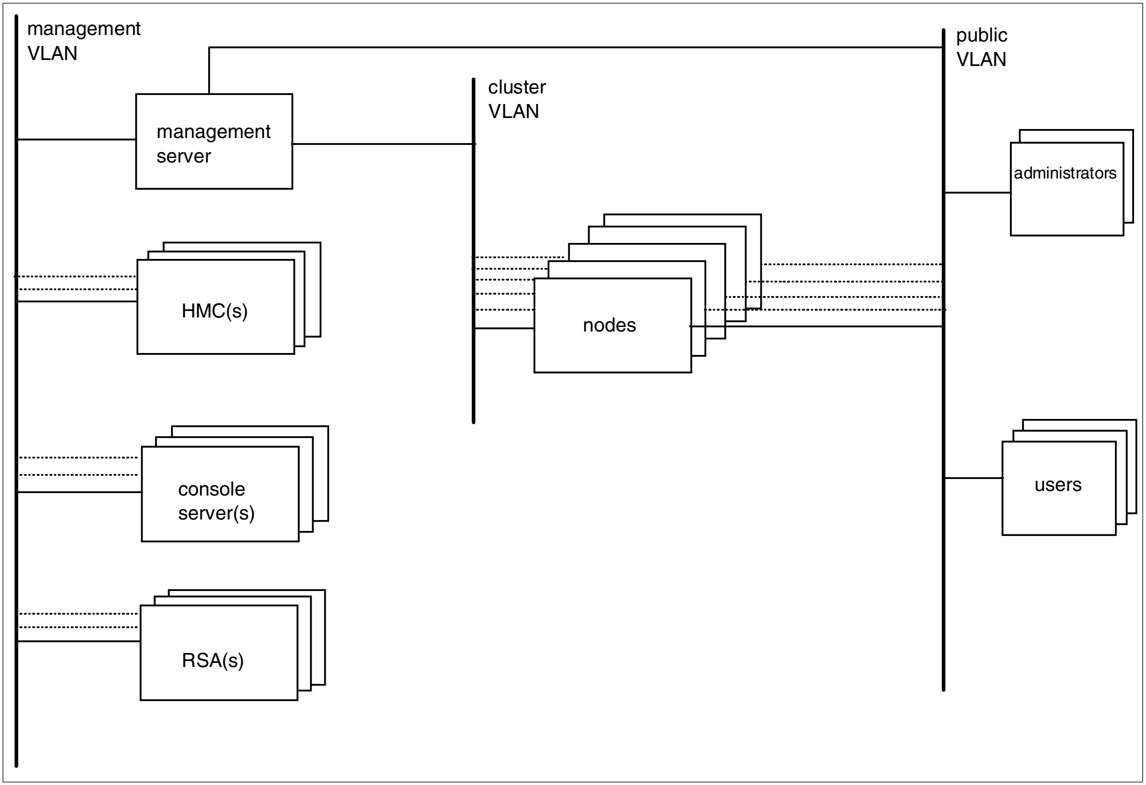


Figure 4-1 Simple network configuration

4.2 Shell security (required parameters)

CSM uses a distributed shell program (**dsh**) to issue commands from the management server to the nodes. By default, the **dsh** shell calls **rsh** to run commands on the nodes. If you require a higher level of security than **rsh** provides, you should consider replacing it with OpenSSH or an alternative secure shell. See Chapter 5, "Securing remote command execution" on page 55 for details of how to do this.

4.3 Configuration file manager (CFM)

Configuration file manager (CFM) is the file distribution functionality provided by CSM. CFM simplifies the task of maintaining common files on multiple managed nodes. The CFM file repository, /cfmroot, is the directory on the CSM server where the common files are stored. Changes to files stored in /cfmroot are pushed out to the nodes from the management server.

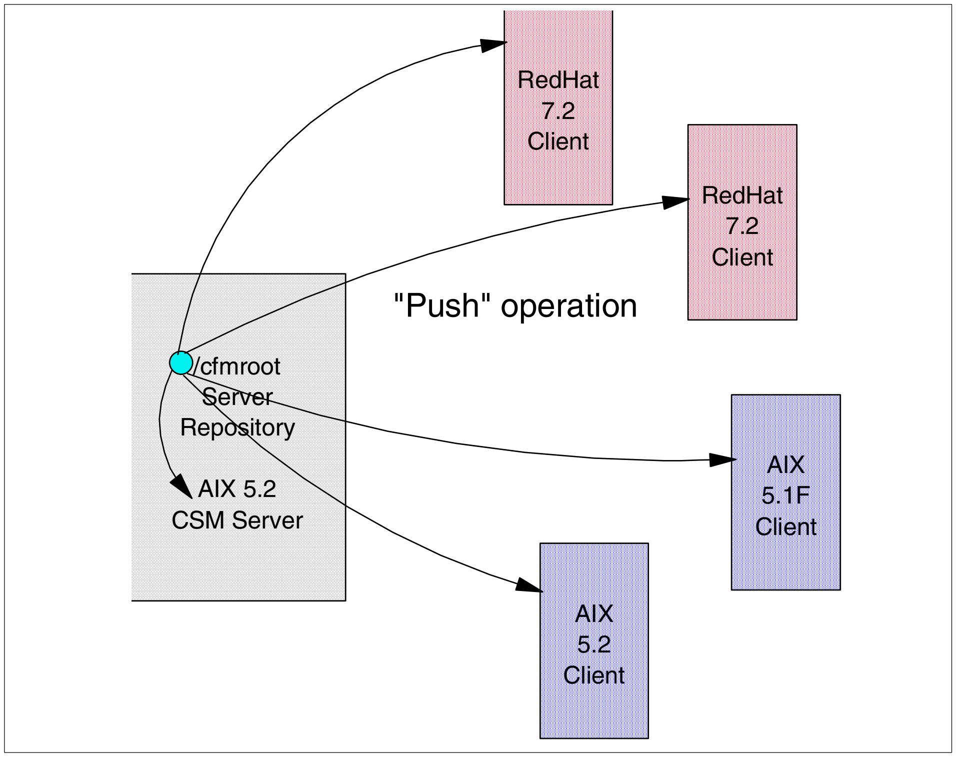


Figure 4-2   Configuration file manager functional diagram

Figure 4-2 illustrates the CFM functionality.

The path name of the configuration file on each node is the same as the path to the server file repository, without /cfmroot. For example, the final destination of a configuration file on the management server called /cfmroot/etc/security is /etc/security on the node.

Note: Files transferred by CFM retain their original modification time stamp and permissions from the management server.

The **cfmupdatenode** command can be issued on the management server to immediately distribute the configuration files located in /cfmroot. User and group ownership are important for CFM:.

At the time of writing, the behavior is as follows:

- For files that are not owned by the root user ID, the user and group names and the UID and GID numbers must be the same across the management server and all nodes to which the file is to be transferred. If all of user name, group name, UID, and GID are not the same on all nodes, the file will not be transferred to any node.
- If the file owner is root, and the group owner is root's group (normally, the *system* group on AIX and the *root* group on Linux), it will be distributed. If the file is owned by the root user, but not root's group, it will be distributed if the group name and GID are the same on all target systems.

A change being developed will cause the behavior to be as follows: If a file is owned by root, it will be distributed regardless of the group ownership. If a file's group does not exist on a destination node, it will be changed to root's group (root on Linux and system on AIX).

4.4 User management

When a cluster grows larger than a few nodes, or larger than a small number of users, it can become difficult and time consuming to manage user IDs, passwords, and groups effectively and securely. Some of the issues that arise include:

- Defining and maintaining the same user IDs and groups on every node. This is time consuming and error prone if done manually. It is useful, if doing this, to use the same UIDs and GIDs on every node.
- It is good practice to change passwords regularly. This is time consuming for users (particularly administrators) who have many user IDs.
- In client/server-based applications, multiple clients frequently need to use user IDs or passwords, or both, to connect to the same server. The passwords for these user IDs must be changed at the same time on each node.

To reduce the impact of these issues, many system administrators choose to synchronize user IDs, UIDs, passwords, groups, and GIDs across some, or all, of the nodes in their cluster. Some of the disadvantages of doing this include:

- It can affect the security of the cluster. If a user ID is compromised on one node, it is compromised on other nodes using the same user ID.

- Passwords should only be reset on the management server. If a user changes a password on a node, it will be overwritten when the files are copied from the management server. You could replace /usr/bin/passwd on every node with a script that tells the user to change the password on the management server, but you need to remember to put the script back if you apply fixes that replace it.

4.5 Security in a heterogeneous environment

In a mixed cluster, there are some security considerations that we must address to avoid potential problems. CFM is designed to distribute files from the management server to all cluster nodes using the **cfmupdatenode** command. This is a convenient and fast method of replicating files across the cluster. However, it can cause problems in your cluster if a file with the wrong format is accidently propagated as a node. The systems administrator must understand and pay close attention to their environment to avoid potential disaster.

An example of this is the /etc/passwd file. The /etc/passwd file does not have the same format in AIX as in Linux. If you accidently use CFM to distribute the AIX /etc/passwd to your Linux nodes, you will not be able to log back on to the Linux nodes. The same problem will occur if you copy the Linux /etc/passwd file into AIX nodes.

We suggest the following approach:

1. Create all AIX user IDs and groups on the management server.
2. Choose a Linux node to be your *Linux master* node. Create all Linux user IDs and groups on this node. Create user IDs with the same UIDs as the management server. Create groups with the same GIDs as are on the management server.
3. Create links to the AIX security files with the _AIXNodes extension in /cfmroot and use CFM to synchronize the security files across the members of the AIXNodes group.
4. Use Network File System (NFS) to mount /etc on the Linux master node onto the management server, copy the Linux security files to /cfmroot with _LinuxNodes extensions, and use CFM to transfer them to the members of the LinuxNodes group.
5. Use Distributed Command Execution Manger (DCEM) or **dsh** to create a home directory for each user ID on each node.

This approach synchronizes user IDs and passwords on all nodes in the cluster and creates home directories on each node for each user ID.

4.6 Web-Based System Manager

Web-Based System Manager, commonly known as WebSM, is a systems management application that can be used to administer AIX servers. In AIX 5L Version 5.2, WebSM contains CSM application plug-ins that provide an interface for monitoring and managing one or more CSM for AIX clusters. The CSM plug-ins can be used to manage a mixed cluster running both Linux and AIX nodes.

WebSM must be configured on all of the servers on which you plan to use it. For detailed instructions about how to install and configure WebSM, see the *AIX 5L Version 5.2 Web-based System Manager Administration Guide*. It is available online at:

http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/wsmadmn/wsmadmn.pdf

To start the WebSM on a CSM management server, run the following command:

```
/usr/bin/wsm
```

The WebSM, which is included with AIX 5L Version 5.2, automatically detects a CSM cluster. Figure 4-3 on page 44 shows the initial WebSM screen with the CSM cluster objects at the bottom of the Navigation Area.

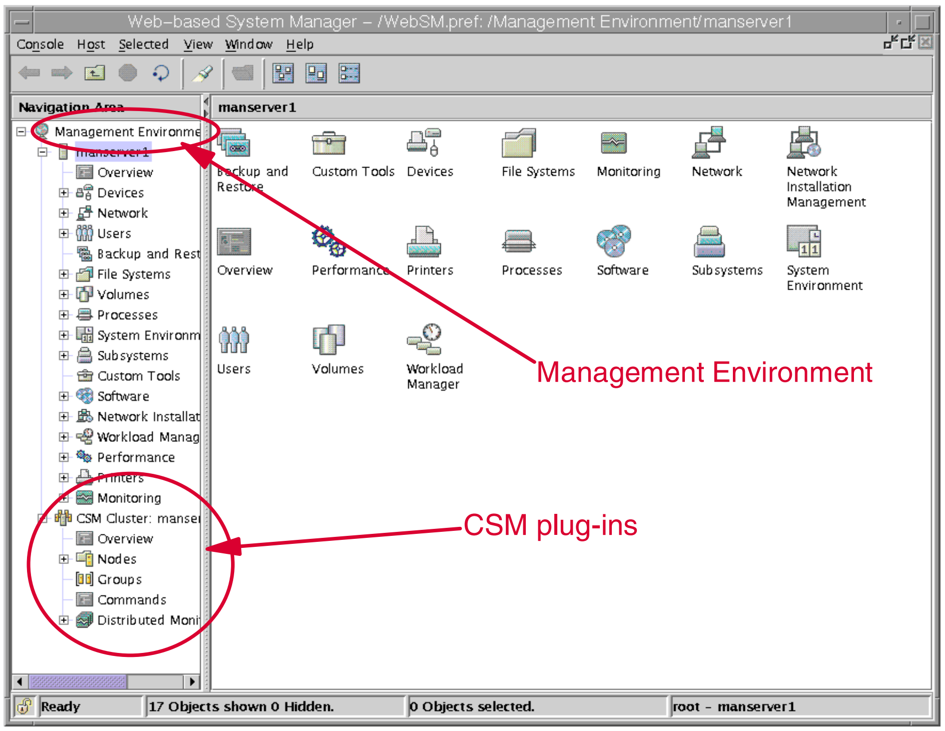


Figure 4-3 Configuration file manager plug-ins for WebSM

WebSM can run in four different modes:

- Stand-alone application mode

  The systems administrator logs on to the server to be administered, runs the **wsm** command to start WebSM, and uses it to perform administrative tasks on that server. No configuration is necessary to run WebSM in stand-alone mode.

- Client/server mode

  The client is the machine that runs the **wsm** command. The server machine is the machine that is administered by the client machine. The **wsm** command runs on the client. The administrator must add the hostname or IP address of the server machine to the WebSM Management Environment on the client machine (see Figure 4-3).

WebSM on the server machine must be configured to allow remote management by running the **`wsmserver -enable`** command. WebSM on the client machine prompts for a user ID and password that it uses to log on to the server machine. Administrative tasks are run on the server machine using that user ID.

When using client/server mode to manage several machines, it is possible to use one server as a managing server (not to be confused with the CSM management server) by running the **`wsm`** command with the **`-host`** option, for example, **`wsm -host mgtserver1`**. To use WebSM in this way, a Web server must be installed on the managing server. The Web server must be configured (using the **`configassist`** command) to support WebSM. The IBM HTTP server is available on the AIX 5L Version 5.2 Expansion Pack.

- Applet mode

  The administrator points his or her browser to the Web server running on a server configured for WebSM (as with client/server mode), for example:

  ```
  http://mgmtserver1/wsm.html
  ```

  The managing server communicates with the machines to be managed. The browser client only communicates with the managing server.

  Servers to be managed using the applet mode must all run the same version of WebSM as the managing server. For security reasons, applets are restricted to loading Java classes only from the Web server that is running the applet. Therefore, the only WebSM Java class files that can be loaded are the class files installed on the managing server.

  WebSM prompts for a user ID and password on the target system. Tasks are run on the target system using that user ID.

- Remote client mode

  The user installs and runs a Java-based remote client package on their Windows or Linux client. This is very similar to client/server mode, because the package provides a **`wsm`** command for Windows or Linux.

  When this WebSM client is run, the user is prompted for a user ID and password on any target system they want to manage. Tasks are run on the target system using that user ID.

  When using this mode, the machine running the Java client communicates directly with the target system.

  The Java remote client is usually installed over the network from an AIX server running a Web server that is configured (using the **`configassist`** command) to serve the package.

Figure 4-4 shows the data flow in the client/server, applet, and remote client modes. In a CSM cluster, the management server makes a good managing server because the WebSM plug-ins support cluster-level administration tasks, and, because it can communicate with all the nodes, node-level administration can also be done. Node-level administration using the Java client requires a network route between the machine running the Java client and the nodes.

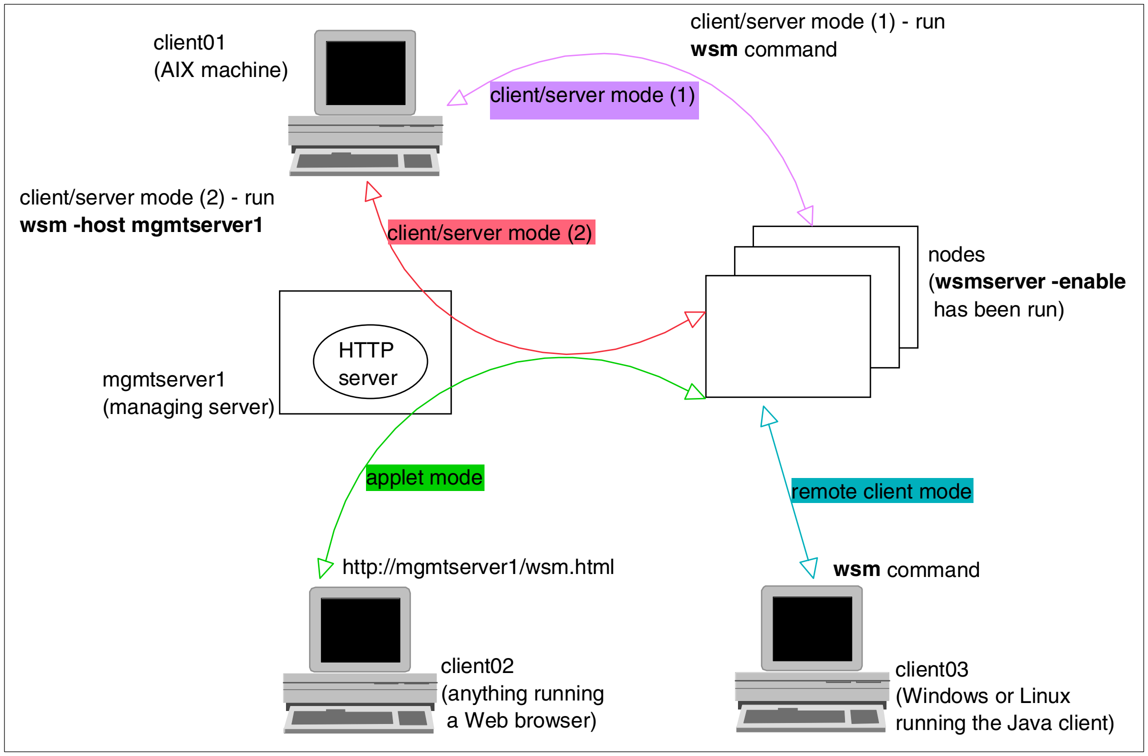

client01
(AIX machine)
client/server mode (1) - run
wsm command
client/server mode (1)
client/server mode (2) - run
wsm -host mgmtserver1
client/server mode (2)
nodes
(**wsmserver -enable** has been run)
HTTP server
mgmtserver1
(managing server)
applet mode
remote client mode
http://mgmtserver1/wsm.html
wsm command
client02
(anything running a Web browser)
client03
(Windows or Linux running the Java client)

Figure 4-4 WebSM configuration modes

4.6.1 Securing Web-Based System Manager

WebSM Security provides for the secure operation of WebSM in each of the three remote modes: client/server, applet, and remote client. In the WebSM secure operation, the managed machines (CSM management servers and nodes) are servers, and the managing users (the system administrators) are the clients.

The communication between the servers and clients uses the SSL protocol. SSL provides server authentication, data encryption, and data integrity. An administrator must enter a user ID and password in order to manage a cluster or a node. The user ID and password are transmitted securely using SSL.

A certificate authority (CA) is required in order to use SSL. A CA is a server that is trusted by other servers to issue signed keys (certificates). You have the option of either using an external CA, or nominating one of the servers in your environment to act as a CA.

Each WebSM server must be configured with the following:

- ► Its own unique private key
- ► A public key signed by the CA

The WebSM client has a file that contains the public keys (ring of keys) of the servers it administers.

When using applet mode, the client must be sure that the applet (.class files) arriving at the browser are coming from the intended server. In this mode, the public key ring file resides on the managing server, not on the workstation that runs the browser. The public key is transferred to the client with rest of the applet .class files. This is because, for security reasons, the browser does not allow applets to read local files.

For sender authentication and integrity of these files, the client must contact the server only using the HTTPS protocol (https://mgmtserver1/wsm.html). In order to make HTTPS available on the managed servers, you can either configure SSL on the Web server on each managed machine, or you can use the SMGate daemon installed with WebSM Security. The SMGate daemon serves as an SSL gateway between the client browser and the Web server.

Note: WebSM Security provides both a graphical interface and command line interface to configure the secure administration

There are two WebSM Security filesets on the AIX 5L Version 5.2 Expansion Pack. The first fileset, systmgt.websm.security, must be installed on each server to be managed using WebSM. The second security fileset, sysmgt.websm.security-us, includes a 128-bit encryption algorithm (however, this is only available in certain countries).

Each server to be managed must also must have a copy of the public key file of the CA in the /usr/websm/codebase directory.

Figure 4-5 on page 48 shows the secured traffic flow in client/server, applet, and remote client modes. The traffic flow is the same as before, except that the SSL protocol is used. The Web server running on the management server has been configured for SSL, and it runs the SMGate daemon. Each managed node has a key file, as do the AIX and Java client machines.

For detailed instructions about how to install and configure WebSM Security, see *AIX 5L Version 5.2 Web-based System Manager Administration Guide*. It is available online at:

http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/wsmadmn/wsmadmn.pdf

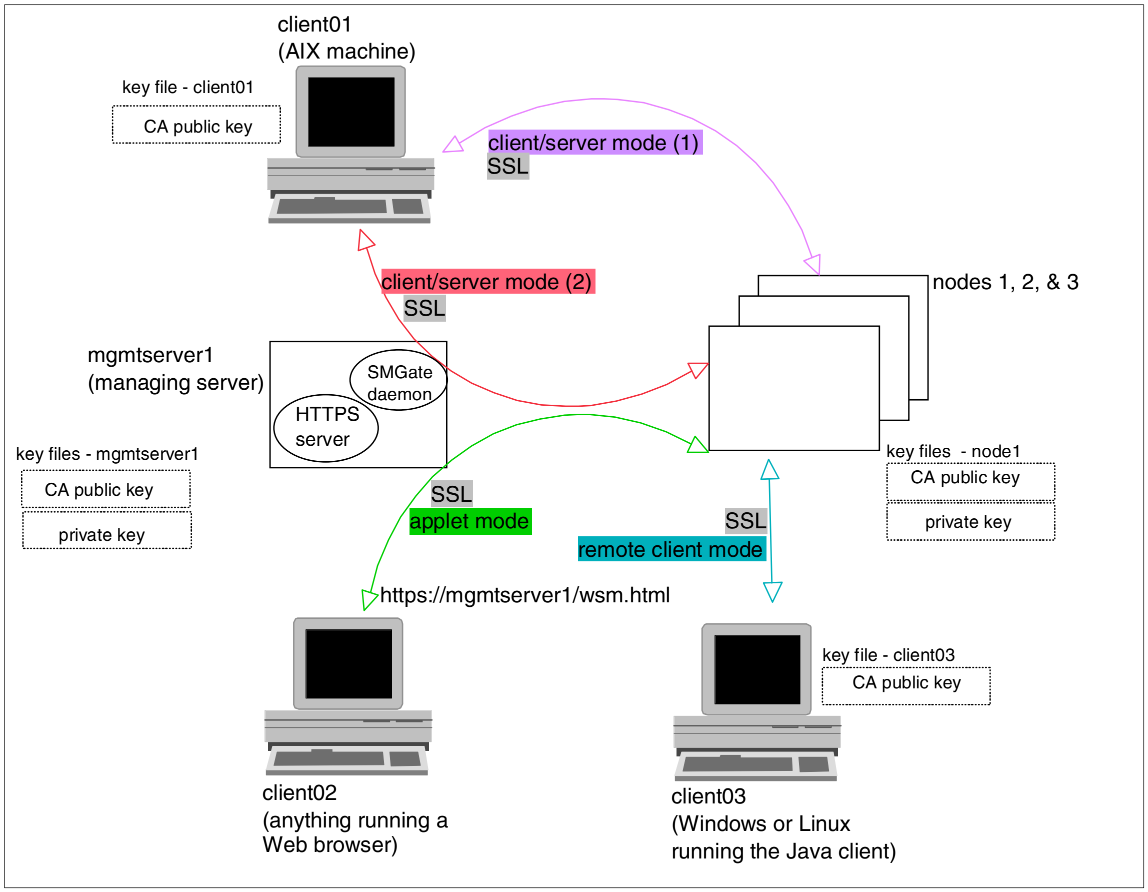


Figure 4-5 Secure WebSM communication with SSL and public and private keys

4.6.2 Installing WebSM Security on a remote client

WebSM Remote Client Security is an additional package that must be installed on Windows or Linux clients that run the WebSM remote client in order to enable Web-based system security. It is usually installed over the network from a server that has been configured to serve downloads of the package.

To install WebSM Remote Client Security on a Windows or Linux client, point your browser to remote_client_security.html on the server you have configured, for example:

```
http://mgmtserver1/remote_client_security.html
```

Download the file and run it on your client. You must also copy your public key file of the CA to the appropriate directory. The directory for Windows clients is C:\Program Files\websm\codebase. The directory for Linux clients is /opt/websm/codebase.

4.7 Security considerations for hardware control

Security controls must be in place to ensure that hardware control commands, such as remote console or remote power, are authenticated and authorized.

4.7.1 User IDs and passwords

HMCs, console servers, and RSAs all require users to authenticate before executing any commands. This includes the CSM management server.

User IDs and passwords for each console server, RSA, and HMC in the cluster are stored in the CSM database. This information can be updated using the **`systemid`** command.

It is good practice to change the passwords on these devices regularly. When you change the passwords, you need to run the **`systemid`** command again to store the new password in the CSM database.

4.7.2 Resource Monitoring and Control access control lists

Commands, such as **`rpower`** (which can power on or off nodes and get their power status) and **`lshwinfo`** (which reports on the hardware in a node), use the security functions of RMC to determine who is allowed to run them. The **`rpower`** command determines who is allowed to run it by calling actions on the IBM.NodeHwCtrl resource class. The **`lshwinfo`** command calls actions on the IBM.HwCtrlPoint resource class to determine who is allowed to run it.

Access to the hardware control classes, and to actions on these classes, is controlled by stanzas in the /var/ct/cfg/ctrmc.acls file. See 3.2.6, “Resource Monitoring and Control access control list” on page 31 for more details about the RMC ACL file, or see *IBM Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference*, SA22-7889.

By default, only root on the CSM management server has the authority to perform these actions. Other users on the management server have read-only access that allows them to view attribute data from the resource classes, but not perform hardware control actions.

Example 4-1 shows stanzas for IBM.NodeHwCtrl and IBM.HwCtrlPoint. In this example, the host is the management server. The user IDs root and david have read and write access. All other users on the management server read-only access.

Example 4-1 Sample stanza in the ctrmc.acls file

```
IBM.NodeHwCtrl
        root@LOCALHOST       *       rw
        david@LOCALHOST      *       rw
        LOCALHOST            *       r

IBM.HwCtrlPoint
        root@LOCALHOST       *       rw
        david@LOCALHOST      *       rw
        LOCALHOST            *       r
```

4.7.3 Console server security

The **`rconsole`** command opens a console window for a node. It uses the Conserver open source package to provide support for multiple read-only consoles on a single node.

The main advantages of using the Conserver software are as follows:

- More than one user can access the console simultaneously (one read/write and the others read-only).
- You can scroll back to view messages that have disappeared from view.

It also provides authentication, so it can be used to restrict access to remote console functions.

Conserver.cf file

The conserver.cf file is the configuration file for the Conserver software. It is read on startup.

The Conserver software restricts connections from clients based on the host access section of the conserver.cf file. It authenticates users against its conserver.passwd file.

By default, CSM creates a conserver.cf file that allows consoles to be started only from the management server. However, the default conserver.passwd file allows any user on the management server to start a console.

If you modify the conserver.cf file, you must restart the Conserver process using the **rconsolerefresh** command. CSM runs the both the **chrconsole** command and the **rconsolerefresh** command whenever a node is added to or removed from the cluster. The **chrconsole** command updates the conserver.cf file with the appropriate changes. The **rconsolerefresh** command forces the daemon to reread the file.

Example 4-2 shows the conserver.cf file used in our environment. There are two nodes defined: lpar1 and lpar2. The management server, manserver1, is allowed to run the **rconsole** command, while testserver1 is not allowed to run the command.

Example 4-2 The conserver.cf file

```
#
# $Id: conserver.cf,v 1.4 2001-06-28 10:24:01-07 bryan Exp $
#
# The character '&' in logfile names are substituted with the console
# name.  Any logfile name that doesn't begin with a '/' has LOGDIR
# prepended to it.  So, most consoles will just have a '&' as the logfile
# name which causes /var/consoles/<consolename> to be used.
#
LOGDIR=/var/log/consoles
TIMESTAMP=1da
#
# list of consoles we serve
#    name : tty[@host] : baud[parity] : logfile : [mark-interval(m|h|d|l)][+]
#    name : !host : port : logfile : [mark-interval(m|h|d|l)][+]
#    name : |command : : logfile : [mark-interval(m|h|d|l)][+]
#
lpar1:|/opt/csm/bin/rconsole -t -c -n lpar1@manserver1::&
lpar2:|/opt/csm/bin/rconsole -t -c -n lpar2@manserver1::&
%%
#
# list of clients we allow
# {trusted|allowed|rejected} : machines
#
allowed: manserver1
rejected: testserver1
```

Conserver.passwd file

The conserver.passwd file is the user authentication and authorization file for the Conserver software. It contains the list of user IDs that are allowed to access consoles.

Example 4-3 shows an example of the conserver.passwd file. Each line contains three fields.

The first field is the user name. The user name field can contain either of the following:

- The logon name of an authorized user
- The string *any*, which means any user

The second field can contain either of the following:

- The encrypted password
- The string *passwd*, which indicates that Conserver should look up the user's password in the operating system's passwd file

The third field can contain either of the following:

- The list of consoles to which the user is permitted to connect
- The string any to allow access to any console

In this file, the root user ID is allowed to start a console for any server. The root user ID password is validated against the password stored in the operating system's passwd file. The user1 user ID is allowed start a console only on lpar1, and its encrypted password has been copied from the /etc/security/passwd file into the conserver.passwd file.

Example 4-3 Sample conserver.passwd file

```
root:*passwd*:any
user1:r71mXjfRGR8rc:lpar1
user2:*passwd*:lpar2
user3:*passwd*:lpar3
user4:*passwd*:any
user5:*passwd*:lpar1
```

By default, CSM creates a conserver.passwd file that allows any user to start a console on any server. You can choose to restrict the authority to run consoles to the root user ID. If so, we recommend that you use the *passwd* method to tell Conserver to use the operating system password file, instead of copying encrypted passwords from the passwd file into the conserver.passwd file.

4.8 Name resolution

CSM 1.3 for AIX uses host-based authentication (HBA) as its authentication method. Reliable name resolution is critical to ensure correct operation of HBA, and by extension, RMC and CtSec. For example, the trusted host list (THL) file contains the host names of the servers it trusts.

If name resolution is incorrectly configured, or broken, your nodes may not be able to communicate with the hosts in their THL files, thus, authentication will fail.

5

Securing remote command execution

This chapter describes how to configure remote command execution in a CSM cluster. It is possible to replace the underlying **dsh** command (the **rsh**) that provides a minimum security level with different, more secure software. Of course, this software must meet some requirements.

In this chapter, we describe installation and configuration instructions for OpenSSH and provide some recommendations in case you decide to use different software.

5.1 Remote command execution software

CSM provides a tool for distributed command execution from the management server to the nodes. This tool is called *distributed shell (dsh)*.

The distributed shell in the CSM cluster is used to issue remote commands in a distributed manner, from the management server to the cluster nodes, in order to ease system administration work. This tool is not used for internal cluster communication between the management server and the nodes.

For internal communication, such as resource control and monitoring, the Reliable Scalable Cluster Technology (RSCT) layer is used. See Figure 5-1 on page 57.

Remote command execution is needed for administrative actions that have to be executed on some or all nodes in a CSM cluster.

You can also configure a cluster to support node-to-node remote commands, but this is not required. The **dsh** command is contained in the csm.dsh fileset, which is installed with AIX 5L Version 5.2. The CSM software provides tools for setting up remote command execution subsystem from the management server to the nodes.

By default, **dsh** relies on the “classic” **rsh** command for remote execution. Unfortunately, **rsh** provides only a minimum security level. The authorization is based on the .rhosts file stored in user’s home directory. The data exchanged between the management server and the nodes is not encrypted.

You can set up the CSM management server to use another remote command execution software instead of **rsh** in order to improve the security inside the CSM cluster.

The software you choose to replace **rsh** must support the **dsh** command arguments. This means that the command you use instead **rsh** must support the parameters and options that the **rsh** command allows.

Our recommendation is to use the OpenSSH packet instead of the r commands. Therefore, the **dsh** command uses the **ssh** command instead of **rsh**. The overview of communication between the management server and the node is shown in Figure 5-1 on page 57.

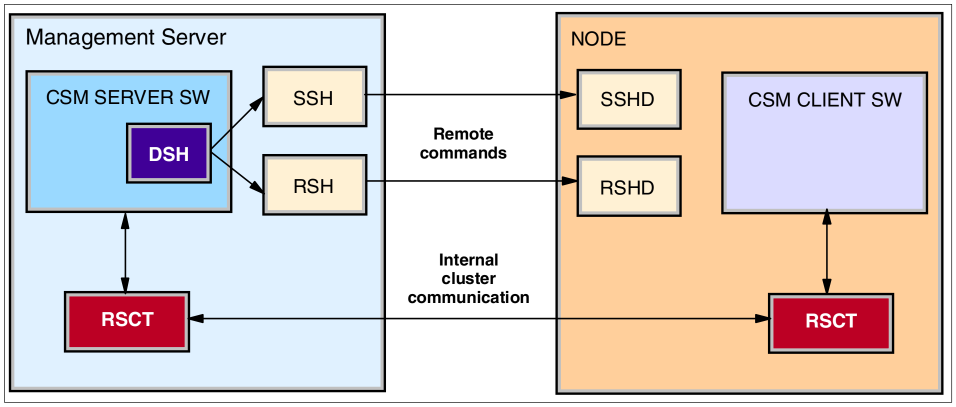


Figure 5-1 Remote command execution in CSM

You can use the **dsh** command on every AIX 5L Version 5.2 machine to issue distributed commands, but only the MS-to-nodes communication is configured automatically. Example 5-1 shows the execution of the **date** command on the nodes called machine1 and machine2.

Example 5-1 The dsh command to listed nodes

```
machine0#> dsh -n machine1,machine2 date
machine1: Mon Nov 4 14:46:47 CST 2002
machine2 Mon Nov 4 14:46:15 CST 2002
```

The **dsh** command communicates with the CSM database. From the database, it retrieves the remote execution shell name to be used, as well as information about the CSM nodegroups membership.

You can use **dsh** to pass commands to the nodegroups. In Example 5-2, the **dsh** command retrieves the remote command execution shell name and the group members for group1 from the CSM database and executes the **date** command on the member nodes.

Example 5-2 The dsh command to nodegroup members

```
MS1#> dsh -N group1 date
node1: Mon Nov  4 15:42:42 CST 2002
node2: Mon Nov  4 15:42:13 CST 2002
node3: Mon Nov  4 15:42:13 CST 2002
```

In our environment, group1 has three members: node1, node2, and node3. Our management server is set up to call the /usr/bin/ssh program for remote command execution whenever a **dsh** command is issued. As a result, the **dsh** command (as in Example 5-2 on page 57) calls the following commands:

```
/usr/bin/ssh -n node1 date
/usr/bin/ssh -n node2 date
/usr/bin/ssh -n node3 date
```

The behavior of the **dsh** command (as in Example 5-2 on page 57) is shown in Figure 5-2.

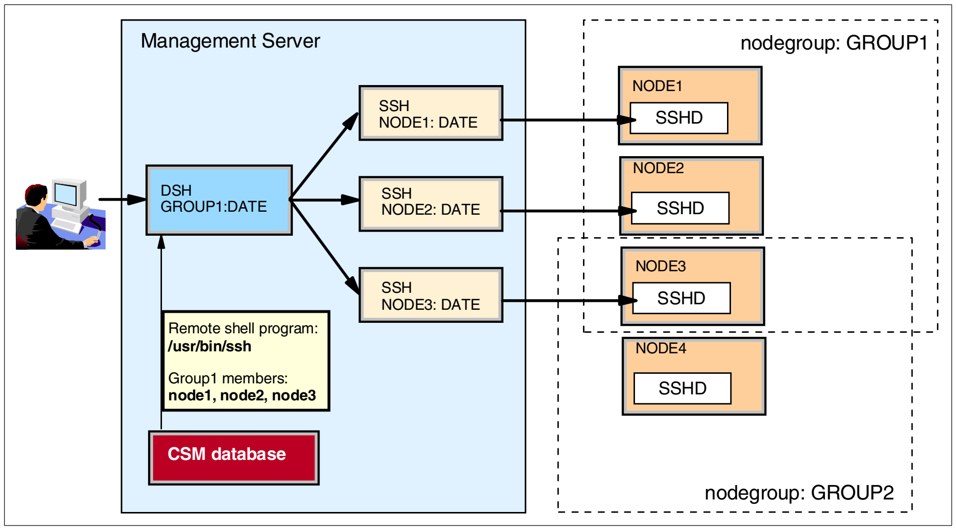


Figure 5-2   Distributed command execution

The remote execution shell program also can be temporarily changed. This can be achieved by changing the environment variable DSH_REMOTE_CMD, as follows:

```
export DSH_REMOTE_CMD=/usr/bin/rsh
```

You can also use the parameter **-r** with the **dsh** command to override both the database setting and the DSH_REMOTE_CMD environment variable:

```
dsh -r /usr/bin/rsh -n node1 date
```

5.2 OpenSSH installation on AIX

In this section, we describe the OpenSSH package installation on the CSM management server and on the nodes. This section also provides instructions for the OpenSSH software configuration.

Note: If you decide to use different software, it may not be possible to use the automated configuration tools we describe in this chapter.

On the CSM management server, you need to install and configure the SSH client software only. On all the nodes the SSH server is required (see Figure 5-1 on page 57).

5.2.1 Downloading OpenSSH and prerequisite OpenSSL software

You have the choice of using the OpenSSH software shipped on the AIX 5L Bonus Pack CD-ROM, or you can download the OpenSSH software from the Internet.

The following list contains some of the Internet locations you can use to obtain the OpenSSH software:

- Linux Toolbox:

  `http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html`

- Cryptographic content for AIX (requires registration):

  `http://www6.software.ibm.com/dl/aixtbx/aixtbx-h?S_PKG=dlaixww&S_TACT=&S_CMP=`

- Open Source Software for AIX:

  `http://oss.software.ibm.com/developerworks/downloads/?group_id=108`

- Bull freeware:

  `http://www.bullfreeware.com/`

In our environment we tested the following OpenSSH software packages:

- `http://www.bullfreeware.com/download/aix43/openssh-3.4.0.0.exe`
- `ftp://www-126.ibm.com/pub/opensshi/3.4p1_51/openssh34p1_51.tar.Z`
- `ftp://www-126.ibm.com/pub/opensshi/3.4p1_52/openssh34p1_52.tar.Z`
- OpenSSH 34p1_52 shipped on the AIX 5.2 Bonus Pack CD-ROM

For the openssh34p1_51.tar.Z and openssh34p1_52.tar.Z packages, as well as for OpenSSH shipped with the AIX 5L Bonus Pack, you also need to install the prerequisite software OpenSSL before you actually install the OpenSSH files.

We used openssl-0.9.6e-2.aix4.3.ppc.rpm from (requires registration):

http://www6.software.ibm.com/dl/aixtbx/aixtbx-i?S_PKG=dlaixww&S_TACT=&S_CMP=

The Bull version of OpenSSH openssh-3.4.0.0.exe does not require OpenSSL, because the OpenSSL libraries are included in this package.

5.2.2 Preinstallation tasks

Although we recommend the use of OpenSSH Version 3.4 for AIX 5L Version 5.2 shipped on the Bonus Pack for AIX 5L Version 5.2 CD-ROM, you can also use other packages. In this section, we describe the steps we performed before the OpenSSH installation.

Prepare the OpenSSL installation file

All OpenSSH versions, except the Bull version, require OpenSSL as prerequisite software. Download the openssl-0.9.6e-2.aix4.3.ppc.rpm file, or a similar one, from the Internet, and store it in a temporary directory on your management server.

We created the /tmp/ssl directory on our management server for this purpose.

To download the openssl-0.9.6e-2.aix4.3.ppc.rpm package, you must first register by filling out the Registration Form found on this page:

http://www6.software.ibm.com/dl/aixtbx/aixtbx-i?S_PKG=dlaixww&S_TACT=&S_CMP=

Prepare the OpenSSH installation files

The steps for this task depend on the OpenSSH software chosen. In this section, we describe the required steps for all the software we tested.

openssh34p1_52 (Bonus Pack)

For OpenSSH Version 3.4 for AIX 5L Version 5.2 shipped on the Bonus Pack for AIX 5L Version 5.2, no preparation action is required.

openssh34p1_52 (Internet download)

Download the OpenSSH Version 3.4 for AIX 5L Version 5.2 from the Internet:

ftp://www-126.ibm.com/pub/opensshi/3.4p1_52/openssh34p1_52.tar.Z

This is a compressed tar file. Complete the following steps before installation:

1. Create a temporary directory. We created the /tmp/ssh directory.
2. Download the file openssh34p1_52.tar.Z to this directory.
3. Change the current directory to the temporary directory. For example:

   ```
   cd /tmp/ssh
   ```

4. Unpack and uncompress the openssh34p1_52.tar.Z file.

   Use the **`zcat openssh34p1_52.tar.Z|tar -xvf -`** command, as shown in Example 5-3.

Example 5-3 Extracting the openssh34p1_p2.tar.Z file

```
mgmtserver2[/tmp/ssh]#ls
openssh34p1_52.tar.Z
mgmtserver2[/tmp/ssh]#zcat openssh34p1_52.tar.Z|tar -xvf -
x openssh.base, 1845248 bytes, 3604 media blocks.
x openssh.license, 641024 bytes, 1252 media blocks.
x openssh.man.en_US, 99328 bytes, 194 media blocks.
x openssh.msg.CA_ES, 17408 bytes, 34 media blocks.
x openssh.msg.CS_CZ, 17408 bytes, 34 media blocks.
..... >>>> Omitted Lines <<<<< .....
x openssh.msg.sk_SK, 16384 bytes, 32 media blocks.
x openssh.msg.zh_CN, 12288 bytes, 24 media blocks.
x openssh.msg.zh_TW, 13312 bytes, 26 media blocks.
mgmtserver2[/tmp/ssh]#
```

openssh34p1_51

For the openssh34p1_51.tar.Z file, the following steps need to be performed before installation:

1. Create a temporary directory. We created the /tmp/ssh directory.
2. Download the file openssh34p1_51.tar.Z to this directory.
3. Change to the temporary directory:

   ```
   cd /tmp/ssh
   ```

4. Unpack and uncompress the openssh34p1_52.tar.Z file.

   Use the **`zcat openssh34p1_51.tar.Z|tar -xvf -`** command, as shown in Example 5-4 on page 62.

Example 5-4 Extracting the openssh34p1_p2.tar.Z file

```
mgmtserver2[/tmp/ssh]#ls
openssh34p1_51.tar.Z
mgmtserver2[/tmp/ssh]#zcat openssh34p1_51.tar.Z |tar -xvf -
x openssh.base, 1865728 bytes, 3644 media blocks.
x openssh.license, 632832 bytes, 1236 media blocks.
x openssh.man.en_US, 99328 bytes, 194 media blocks.
x openssh.msg.CA_ES, 17408 bytes, 34 media blocks.
x openssh.msg.CS_CZ, 17408 bytes, 34 media blocks.
x openssh.msg.DE_DE, 17408 bytes, 34 media blocks.
...........>>>>>>Omitted lines <<<<<<........
x openssh.msg.pl_PL, 16384 bytes, 32 media blocks.
x openssh.msg.pt_BR, 16384 bytes, 32 media blocks.
x openssh.msg.ru_RU, 16384 bytes, 32 media blocks.
x openssh.msg.sk_SK, 16384 bytes, 32 media blocks.
x openssh.msg.zh_CN, 12288 bytes, 24 media blocks.
x openssh.msg.zh_TW, 13312 bytes, 26 media blocks.
mgmtserver2[/tmp/ssh]#
```

5. Create an empty /.ssh/prng_seed file:

   ```
   mkdir /.ssh
   touch /.ssh/prng_seed
   ```

Important: If you plan to use the Network Installation Manager (NIM) for the OpenSSH installation, do not use this version. The /.ssh/prng_seed file must be created prior the installation, and it is very complicated to implement it in NIM.

openssh-3.4.0.0.exe (Bull)

For the openssh-3.4.0.0.exe file, the following steps need to be performed before installation:

1. Create a temporary directory. We created the /tmp/ssh directory.
2. Download the file openssh-3.4.0.0.exe to this directory from:

   `http://www.bullfreeware.com/download/aix43/openssh-3.4.0.0.exe`

3. Change the current directory to the temporary directory:

   ```
   cd /tmp/ssh
   ```

4. Make this file executable:

   ```
   chmod +x openssh-3.4.0.0.exe
   ```

5. Execute this file to extract the package to the temporary directory:

   ```
   . ./openssh-3.4.0.0.exe
   ```

Example 5-5 Extracting the openssh-3.4.0.0.exe file

```
mgmtserver2[/tmp/ssh]#ls -l
total 5056
-rwxr-x--x   1 root system      2584761 Oct 30 17:49 openssh-3.4.0.0.exe
mgmtserver2[/tmp/ssh]#./openssh-3.4.0.0.exe
UnZipSFX 5.41 of 16 April 2000, by Info-ZIP (Zip-Bugs@lists.wku.edu).
  inflating: openssh-3.4.0.0.bff
  inflating: openssh-3.4.0.0.bff.asc
mgmtserver2[/tmp/ssh]#
```

5.2.3 Installing SSH on AIX manually

You need the SSH server software to run on all nodes managed by CSM. However, the SSH client software is a prerequisite of the SSH server, so you need to install both the SSH server, as well as the client.

On the CSM management server, you need only the SSH client software. The SSH server is not used by the CSM. The package downloaded from the Bull Web site installs both the SSH client and the SSH server.

Complete the following OpenSSH installation steps:

1. Perform the steps required before the installation. See 5.2.2, “Preinstallation tasks” on page 60.
2. Install openssl-0.9.6e-2.aix4.3.ppc.rpm or a similar one (not required for OpenSSH 3.4 downloaded from Bull):
   a. Change to the temporary directory where you stored the OpenSSL fileset. For example:

      ```
      cd /tmp/ssl
      ```

   b. Use **`smit install_latest`** or the following command to install this file:

      ```
      rpm -i openssl-0.9.6e-2.aix4.3.ppc.rpm
      ```

   c. Remove the temporary directory:

      ```
      rm -rf /tmp/ssl
      ```

3. Install the chosen OpenSSH software:
   a. Change to the temporary directory that contains the OpenSSL fileset. For example:

      ```
      cd /tmp/ssh
      ```

   b. Use **`smit install_latest`** to install SSH:
      i. In the first dialog box, enter the /tmp/ssh directory.

   ii. In the second dialog box, change “ACCEPT new license agreements?” to **YES**.

   If you want to install the client SSH software only (for example, on the management server), use the F4 key for the **SOFTWARE to install** field and select only the SSH client software. If you want to install the SSH server, leave the default value **_all_latest**.

Note: If you are installing the Bull version of OpenSSH, you may get some errors:

```
Disabling protocol version 2. Could not load host key
Privilege separation user sshd does not exist
rc.openssh: CMD: error detected in ....
```

This message can be ignored for now. This issue will be addressed in 5.2.4, “Post-installation tasks” on page 64.

c. After the installation is finished, remove the temporary directory:

```
rm -rf /tmp/ssh
```

4. Perform the post-installation tasks described in 5.2.4, “Post-installation tasks” on page 64.

5.2.4 Post-installation tasks

Depending on the version of OpenSSH software, several actions may be needed to make the OpenSSH software functional. This section describes these actions for each version we used in our environment.

openssh34p1_52 (Bonus Pack and Internet download)

After the OpenSSH Version 3.4 for AIX 5L Version 5.2 server software installation, we performed the following steps:

1. Create the /etc/pam.conf file with the contents shown in Example 5-6.

Example 5-6 Recommended contents of the /etc/pam.conf file

```
sshd    auth            required        /usr/lib/security/pam_aix
OTHER   auth            required        /usr/lib/security/pam_aix
sshd    account         required        /usr/lib/security/pam_aix
OTHER   account         required        /usr/lib/security/pam_aix
sshd    password        required        /usr/lib/security/pam_aix
OTHER   password        required        /usr/lib/security/pam_aix
sshd    session         required        /usr/lib/security/pam_aix
OTHER   session         required        /usr/lib/security/pam_aix
```

You can use the command shown in Example 5-7 to create this file.

Example 5-7 Command for /etc/pam.conf creation

```
cat <<EOF >/etc/pam.conf
sshd    auth            required        /usr/lib/security/pam_aix
OTHER   auth            required        /usr/lib/security/pam_aix
sshd    account         required        /usr/lib/security/pam_aix
OTHER   account         required        /usr/lib/security/pam_aix
sshd    password        required        /usr/lib/security/pam_aix
OTHER   password        required        /usr/lib/security/pam_aix
sshd    session         required        /usr/lib/security/pam_aix
OTHER   session         required        /usr/lib/security/pam_aix
EOF
```

2. Update the /etc/inittab file to start the sshd daemon automatically after reboot:

   ```
   mkitab -i rctcpip "sshd:23456789:once:startsrc -s sshd >/dev/console 2>&1"
   ```

3. Start the sshd daemon (the OpenSSH server):

   ```
   startsrc -s sshd
   ```

For openssh34p1_51

We performed the following steps:

1. Update the /etc/inittab file to start the sshd daemon automatically after reboot:

   ```
   mkitab -i rctcpip "sshd:23456789:once:startsrc -s sshd >/dev/console 2>&1"
   ```

2. Start the sshd daemon, the OpenSSH server:

   ```
   startsrc -s sshd
   ```

For openssh-3.4.0.0.exe (Bull)

During the installation of this package, some errors may be reported (see 5.2.3, “Installing SSH on AIX manually” on page 63). The following steps describe how to fix these errors and start the SSH server on the node:

1. Create the sshd user:

   ```
   mkuser sshd
   ```

2. Create the /var/empty directory:

   ```
   mkdir /var/empty
   ```

3. Modify the value of the **-h** parameter for sshd in the /etc/rc.openssh script that is used to start the sshd daemon.

   The **-h** parameter specifies the primary key file of the running sshd (server). Three types of keys are generated automatically during the installation, as shown in Table 5-1 on page 66.

Table 5-1 Key files generated during OpenSSH (Bull) installation

Key type	Private key file name	Public key file name
rsa1	/etc/openssh/ssh_host_key	/etc/openssh/ssh_host_key.pub
rsa	/etc/openssh/ssh_host_rsa_key	/etc/openssh/ssh_host_rsa_key.pub
dsa	/etc/openssh/ssh_host_dsa_key	/etc/openssh/ssh_host_dsa_key.pub

We do not recommend using the rsa1 key type, because rsa and dsa provide a higher security level.

The original value of the **-h** parameter is /etc/openssh/ssh_host_key.

The ssh_host_key file contains the rsa1 type of key. If you want to use the rsa key instead of rsa1, you should change the value of the **-h** parameter to:

```
/etc/openssh/ssh_host_rsa_key
```

Edit the /etc/rc.openssh file and change the value of the **-h** parameter manually, or you can use Example 5-8.

Example 5-8 Modifying the OpenSSH control file

```
str1='-h \/etc\/openssh\/ssh_host_key'
str2='-h \/etc\/openssh\/ssh_host_rsa_key'
sed s/"$str1"/"$str2"/ /etc/rc.openssh >/tmp/rc.openssh
mv /tmp/rc.openssh /etc/rc.openssh
```

4. Ensure that the sshd process is not running (`ps -ef|grep [s]shd`) and start it using the **`/etc/rc.openssh`** script (which will be automatically executed at system init time, during the machine boot):

   ```
   /etc/rc.openssh
   ```

5. Link the /usr/local/bin files to /usr/bin on the management server, because CSM software requires the SSH binaries in the /usr/bin directory for SSH autoconfiguration. You can use the following command:

   ```
   ln -s /usr/local/bin/* /usr/bin/
   ```

5.2.5 Installing OpenSSH 3.4 for AIX 5L on AIX servers using NIM

You can also use the automated procedure offered by NIM to install OpenSSH on your nodes.

This section describes how to set up the NIM server to install OpenSSH Version 3.4 for AIX 5L Version 5.2 on the nodes.

Configure NIM on the CSM management server. See *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859, for more information.

When you create the lpp_source NIM resource, a directory structure similar to the following is created on your system:

```
/csminstall/AIX/aix520/lpp_source/installp/ppc
/csminstall/AIX/aix520/lpp_source/RPMS/ppc
/csminstall/AIX/aix520/lpp_source/usr
```

The steps for adding the SSH server installation to NIM are as follows:

1. Create an installp_bundle resource file.

   The bundle file is a formatted list of the files to be installed by NIM. You can create the file in any directory on the NIM server. We created the `/csminstall/AIX/aix520/sshd34-52.bnd` file, with the content shown in Example 5-9.

Example 5-9 Example of a bundle file for the sshd installation

```
#cat /csminstall/AIX/aix520/sshd34-52.bnd
R:openssl-0.9.6e-2
I:openssh.base.client
I:openssh.base.server
I:openssh.man.en_US
```

Note: You may use a different version of the openssl fileset. In that case, you should modify the bundle file accordingly.

2. Copy the installation filesets listed in the sshd34-52.bnd file to the appropriate lpp_source directories. Copy the installp files into the installp/ppc lpp_source subdirectory and the rpm file to the lpp_source RPMS/ppc subdirectory.

   You can use the **gencopy** command to copy your files to the appropriate directories:

   a. Insert the Bonus Pack CD-ROM.

   b. Use the **gencopy** command to copy the SSH installp files:

   ```
   gencopy -t /csminstall/AIX/aix520/lpp_source/ -d /dev/cd0 \
                         -f /csminstall/AIX/aix520/sshd34-52.bnd
   ```

   c. Use the **gencopy** command to copy the SSL RPM file:

   ```
   gencopy -t /csminstall/AIX/aix520/lpp_source/ -d /tmp/ssl/ \
                         -f /csminstall/AIX/aix520/sshd34-52.bnd
   ```

   The -d parameter specifies the source directory. Use the real location of your openssl-0.9.6e-2*.rpm installation file.

   You can also copy these files manually to the appropriate lpp_source directories, as shown in Example 5-10 on page 68.

Example 5-10 Copy the installation source files to the lpp_source subdirectories

```
#insert the Bonus Pack CD-ROM
mount /cdrom
cd /cdrom/installp/ppc
cp openssh.base.client* /csminstall/AIX/aix520/lpp_source/installp/ppc/
cp openssh.base.server* /csminstall/AIX/aix520/lpp_source/installp/ppc/
cp openssh.man.en_US* /csminstall/AIX/aix520/lpp_source/installp/ppc/
cd /tmp/ssl/ #go to the directory where your openssl resides
cp openssl-0.9.6e-2* /csminstall/AIX/aix520/lpp_source/RPMS/ppc/
```

To verify this step:

```
find /csminstall/AIX/aix520/lpp_source/ -name ‘openss*’
```

The output of the **find** command should be similar to this:

```
/csminstall/AIX/aix520/lpp_source/RPMS/ppc/openssl-0.9.6e-2.aix4.3.ppc.rpm
/csminstall/AIX/aix520/lpp_source/installp/ppc/openssh.base
/csminstall/AIX/aix520/lpp_source/installp/ppc/openssh.man.en_US
```

To finish copying the source files, you have to recreate the .toc file in the installp/ppc subdirectory:

```
cd /csminstall/AIX/aix520/lpp_source/installp/ppc/
rm .toc
inutoc
```

3. Create a script resource file.

   The script resource file is a regular shell script. In later steps, we set up the NIM to execute this script on the target node. You can create the file in any directory on the NIM server. We use this script to create the /etc/pam.conf file that is required for the sshd daemon (in OpenSSH Version 3.4. for AIX 5L Version 5.2) to accept client connections. In this script, we also configure the sshd to start automatically at the node’s initialization time (after reboot). Finally, the sshd subsystem is started.

   We created the /csminstall/AIX/aix520/sshd_conf.ksh file, as shown in Example 5-11 on page 69.

Example 5-11 Script file for the sshd minimum configuration

```
#cat /csminstall/AIX/aix520/sshd_conf.ksh
#!/usr/bin/ksh
echo pam_conf: creating file /etc/pam.conf
[ -r /etc/pam.conf ] || cat <<EOF >/etc/pam.conf
sshd    auth            required        /usr/lib/security/pam_aix
OTHER   auth            required        /usr/lib/security/pam_aix
sshd    account         required        /usr/lib/security/pam_aix
OTHER   account         required        /usr/lib/security/pam_aix
sshd    password        required        /usr/lib/security/pam_aix
OTHER   password        required        /usr/lib/security/pam_aix
sshd    session         required        /usr/lib/security/pam_aix
OTHER   session         required        /usr/lib/security/pam_aix
EOF
echo pam_conf: updating /etc/inittab to start sshd automatically
rmitab "sshd" 2>/dev/null
mkitab -i rctcpip "sshd:23456789:once:startsrc -s sshd >/dev/console 2>&1"
echo pam_conf: starting sshd
stopsrc -s sshd
startsrc -s sshd
```

Note: You can also use this script to modify your sshd_conf file. However, the CSM cluster SSH environment created does not need further configuration at this point.

4. Define the installp_bundle file sshd34-52.bnd as a resource to the NIM.

   You can use **`smit`** to define the NIM resource:

   ```
   smit nim_mkres -> installp_bundle
   ```

   You can also use the **`nim`** command, as shown in Example 5-12 on page 70.

5. Define the script file sshd_conf.ksh to the NIM as a resource.

   You can use **`smit`** to define the NIM resource:

   ```
   smit nim_mkres -> script
   ```

   You can also use the **`nim`** command, as shown in Example 5-12 on page 70.

Example 5-12 Defining the NIM resources for the OpenSSH installation

```
#sshd34-52 installp_bundle resource definition
nim -o remove sshd34-52 2>/dev/null
nim -o define -t installp_bundle -a server=master \
   -a location='/csminstall/AIX/aix520/sshd34-52.bnd' sshd34-52
#sshd_conf script resource definition
nim -o remove sshd_conf 2>/dev/null
nim -o define -t script -a server=master \
   -a location='/csminstall/AIX/aix520/sshd_conf.ksh' sshd_conf
```

After the NIM server is configured for the SSH installation, you can start the installation using the NIM as described in the following steps:

1. Allocate the resources to the node to be installed.

   Do not forget to allocate the lpp_source, sshd34-52, and sshd_conf resources.

   You can use **smit** to allocate the resources to a node or a group of nodes:

   ```
   smit nim_mac_res
   smit nim_grp_res
   ```

2. Install the software on the node. Use the cust NIM operation for a partial installation or the bos_inst NIM operation for a complete node installation.

   You can use **smit** to start the installation of the node or group of nodes:

   ```
   smit nim_mac_op
   smit nim_grp_op
   ```

5.2.6 Verifying the SSH installation on the AIX nodes

Use the **lslpp** command to list SSH filesets:

```
lslpp -l|grep ssh
```

The output should contain the following filesets:

- For the Bull package:
  ```
  freeware.openssh.rte      3.4.0.0  COMMITTED  Openssh 3.4 p1
  ```
- For the 3.4p1 version:
  ```
  openssh.base.client       3.4.0.0  COMMITTED  Open Secure Shell Commands
  openssh.base.server       3.4.0.0  COMMITTED  Open Secure Shell Server
  openssh.msg.en_US         3.4.0.0  COMMITTED  Open Secure Shell Messages -
  openssh.base.client       3.4.0.0  COMMITTED  Open Secure Shell Commands
  openssh.base.server       3.4.0.0  COMMITTED  Open Secure Shell Server
  ```

Check if sshd is running:

```
ps -ef|grep [s]shd
```

If it is not running, start the sshd daemon manually.

You can use the line in /etc/inittab as an example of how to start the sshd daemon:

```
grep ssh /etc/inittab
```

To start the sshd daemon:

- For the Bull version:

  ```
  rcossh:2:once:/etc/rc.openssh >/dev/console 2>&1
  ```

- For the 3.4p1 version:

  ```
  sshd:23456789:once:startsrc -s sshd >/dev/console 2>&1
  ```

At this time, you should be able to connect from the management server to the node using **`ssh`** with password authentication:

```
ssh -n node1 date
```

You will be asked to accept node1 as your *known host*. If you accept it, you will be asked for the root password. Finally, the **`date`** command is executed on node1.

5.3 Installing SSH on Linux nodes

Linux nodes have the OpenSSH package as part of the base server bundle installation (RedHat 7.2 and 7.3). The SSH package version shipped with the Linux operating system can be used within the CSM. You can also download the OpenSSH software from the Internet.

To verify the installation, list the OpenSSH packages:

```
rpm -aq|grep openssh
```

Ensure that the sshd daemon is running on your Linux node:

```
ps -ef|grep sshd
```

If it is not installed or not running on the node, follow the Linux documentation to fix the problem.

Try to connect to the Linux node using the SSH client from the management server. You should be able to connect as the root user to the Linux node using password authentication.

5.4 OpenSSH configuration inside the CSM cluster

This section describes how to configure OpenSSH (Secure Shell) to work within a CSM cluster and how to configure the CSM software to use the **ssh** command provided by the OpenSSH software, instead of using **rsh**.

By the end of this section:

- The **dsh** command should be set up to use the **ssh** command instead of **rsh**.
- The **ssh** command should run properly to all nodes within a cluster.
- The **ssh** command should not request a password or other user intervention.

5.4.1 Preliminary actions

Before the actual configuration steps, you should do the following:

1. Ensure that the sshd process is running on all nodes.

   This can be verified by command:

   ```
   ps -ef|grep [s]shd
   ```

   At least one sshd process should run on each node.

   If there is no running sshd, correct this, as shown in 5.2.6, "Verifying the SSH installation on the AIX nodes" on page 70.

2. On the management server, ensure that the **ssh**, **ssh-keygen** and **scp** commands are in the /usr/bin directory and executable by root.

   If they are in another directory, you should create a link to /usr/bin directory:

   ```
   ln -s <source_file> <link_name>
   ```

   In this example, SSH binaries are the `/usr/local/bin/ssh` directory, and the **ln** command is used to create the soft links in the /usr/bin directory:

   ```
   ln -s /usr/local/bin/ssh /usr/bin/ssh
   ln -s /usr/local/bin/ssh-keygen /usr/bin/ssh-keygen
   ln -s /usr/local/bin/scp /usr/bin/scp
   ```

Important: If the files are not placed on the /usr/bin/ directory, the automated SSH configuration process will fail.

5.4.2 Update the Cluster Systems Management database

The first configuration step is to update the CSM database with the name of the program to be used for remote execution (by **dsh**), that is **/usr/bin/ssh**. By default, the **/usr/bin/rsh** command is set up.

You can check the current setting using the **csmconfig** command:

```
# csmconfig|grep Shell
RemoteShell = /usr/bin/rsh
SetupRemoteShell = 1
```

To change this item in the database to /usr/bin/ssh:

```
csmconfig -r /usr/bin/ssh SetupRemoteShell=1
```

The **-r** parameter and its value /usr/bin/ssh tells CSM to use /usr/bin/ssh for distributed commands.

A SetupRemoteShell attribute value of 1 means that CSM management server configures a remote shell on nodes automatically.

Note: Alternatively, you can use the **installms** command with parameters:

```
   RemoteShell=/usr/bin/ssh
   SetupRemoteShell=1
```

This command takes more time than the **csmconfig** command. It not only configures the remote shell, but also installs the available CSM packages. For more information, refer to *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859.

For example:

```
   installms -x RemoteShell=/usr/bin/ssh SetupRemoteShell=1
```

To verify, use the **csmconfig** command, without any parameters:

```
# csmconfig|grep Shell
RemoteShell = /usr/bin/ssh
SetupRemoteShell = 1
```

5.4.3 Checking the dsh settings

The distributed shell command **dsh** calls a remote shell program to execute commands on the nodes. The name of the program to be used by **dsh** is stored in the CSM database attribute RemoteShell.

The default value stored in the CSM database, used for remote command execution, can be overridden either by setting an environment variable or by the **-r** flag for the **dsh** command.

This is the order of precedence for the remote execution program:

- The **-r** flag of the **dsh** command
- The DSH_REMOTE_CMD environment variable

- The CSM database RemoteShell attribute

In this example, the **rsh** command will be called by the **dsh** command:

```
#csmconfig|grep Shell
RemoteShell = /usr/bin/ssh
SetupRemoteShell = 1

#echo $DSH_REMOTE_CMD
/usr/bin/rsh
```

Before you start configuring SSH, ensure that the DSH_REMOTE_CMD environment variable is not set. To ensure this, use the following command on the CSM management server:

```
unset DSH_REMOTE_CMD
```

5.4.4 Set up OpenSSH

If you are using the OpenSSH software, you do not need to configure your management server and the nodes manually. You can use the **updatenode** command.

If the node is already added to the cluster, its mode is recognized as *Managed* and is automatically assigned to the ManagedNodes nodegroup. If the node is already defined to the CSM, but not configured, its mode is *PreManaged*.

You can display the mode of the node by using the **lsnode** command:

```
lsnode -a Mode
```

If the node is in PreManaged mode, add a new node by following the steps in *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859. The SSH is configured automatically for this node by the **updatenode** command.

If the node is in Managed mode, the node is already configured into the cluster. You need to you use the **updatenode** command with **-k** flag to force the SSH configuration process. The following example explains how to set up or reconfigure the SSH managed node1:

```
updatenode -k -n node1
```

To configure OpenSSH on all the nodes in Managed mode, issue:

```
updatenode -k -N ManagedNodes
```

The **-k** flag can be used with the **updatenode** command even for the nodes in the PreManaged mode. This example sets up the Secure Shell on all Managed, as well as PreManaged, nodes:

```
updatenode -k -a
```

Important: The **-k** flag of the **updatenode** command forces the key exchange for both RSCT and SSH between the management server and the node. This posses a security threat if your network is not completely secure. A rogue machine can replace the management server key with its own key and thus have all the privileges of the management server. To avoid this potential problem, you can configure SSH (exchange the keys) manually.

To exchange the SSH keys manually, do the following:

1. Create the /.ssh directory on the node.
2. If you are configuring SSH on the first node in your cluster, you probably also need to generate the management server's key pair (for the SSH client). Use the **ssh-keygen** command, as shown in Example 5-13, to generate the new key pair.

Important: Do this step only if there are no key files on the management server! Do not rewrite the existing file key file, or the management server will not be able to communicate with the previously configured nodes.

Example 5-13 Generating new keys on the management server

```
ms1# ssh-keygen -t rsa -N '' -f /.ssh/id_rsa
Generating public/private rsa key pair.
Your identification has been saved in /.ssh/id_rsa.
Your public key has been saved in /.ssh/id_rsa.pub.
The key fingerprint is:
d9:8e:e5:2f:95:81:d2:47:7b:c9:2e:06:21:b8:90:85 root@ms1
```

3. Add the management server's generated public key (/.ssh/id_rsa.pub) to the /.ssh/authorized_keys file on the node. Do not use the network to complete this step. Use a floppy disk or other secure way to deliver this key to the node.
4. Capture the RSA fingerprint of the key pair that is used by the sshd daemon on the node. By default, the key pair used by sshd is stored in the /etc/ssh directory. Use the **ssh-keygen** command on the node, as shown in Example 5-14 on page 76.

Example 5-14 Get the RSA fingerprint of the node

```
node1#ssh-keygen -l -f /etc/ssh/ssh_host_rsa_key
1024 56:5a:a4:49:90:5e:1c:8c:5c:45:9e:01:d5:03:f1:b7 /etc/ssh/ssh_host_rsa_key.pub
```

5. Execute a **date** command on the node, using **ssh** from the management server. See the output shown in Example 5-15. Compare the fingerprint shown in the **ssh** command output with the fingerprint displayed in Example 5-14. If the fingerprints are the same, you can answer yes to accept the node1 as the known host.

Example 5-15 Updating the known_hosts file on the management server

```
ms1[/.ssh]#ssh -n node1 date
The authenticity of host 'node1 (9.12.0.159)' can't be established.
 RSA key fingerprint is 56:5a:a4:49:90:5e:1c:8c:5c:45:9e:01:d5:03:f1:b7.
 Are you sure you want to continue connecting (yes/no)?

>>>>>>> answer “yes” if you verified fingerprint to be authentic <<<<<<

Warning: Permanently added 'node1,9.12.0.159' (RSA) to the list of known hosts.
Tue Nov  5 21:16:14 CST 2002
```

5.4.5 How the automated configuration works

The sshd daemon (SSH server) is running on every node. The **ssh** or **dsh** commands are issued on the management server to execute a command on the node, or nodes.

If everything is configured properly, a public and private key (PPK) pair is used for authentication. The root user password is no longer used for authentication. You can change the root password on the managed nodes and still be able to execute remote commands from the management server, as long as the keys remain unchanged.

On the CSM management server, you need the following:

- SSH client software.
- The configuration file for SSH, for example, /etc/openssh/ssh_conf.
- The known_hosts file:
  - Usually in the /.ssh directory.
  - Contains the public keys of the nodes.
- The private key of the server (management server), for example, id_rsa, id_dsa, or identity file.

On the nodes, you need the following:

- The sshd daemon running.

- The configuration file for sshd, for example, /etc/openssh/sshd_conf.
- A file or files containing the public keys of authorized hosts:
  - /.ssh/authorized_keys and /.ssh/authorized_keys2 for protocol 2 by default.
  - The public key of the management server is also stored in this file.
- The node's private key, for example, id_rsa, id_dsa, or identity file.

The following steps explain how the **updatenode** command configures SSH:

1. If the **updatenode** command does not find the PPK pair on the management server, new keys are generated:
   - The rsa1 key is stored in /.ssh/identity.pub.
   - The rsa key is stored in /.ssh/id_rsa.pub.
   - The dsa key is stored in /.ssh/id_dsa.pub.
2. The public keys are copied to the authorized_keys file (rsa1 public key) and the authorized_keys2 file (rsa and dsa public keys) in the /csminstall/csm/config/.ssh directory.
3. The copy.perl script is created in the /csminstall/csm/config/.ssh/ directory. Later, this script will be run on the node to set up the SSH on that node.
4. At this time, the node's root password is requested. The answer is stored in a temporary variable (in the memory, not on file) for later use.
5. The **updatenode** command calls the **/usr/bin/scp** command (scp uses sshd as well) to copy the /csminstall/csm/config/.ssh/ directory and its contents (authorized_keys, authorized_keys2, and copy.perl files) to /tmp/.ssh on the node. The root password stored in the memory variable is now used for authentication. At this time, the node's public key is automatically downloaded from the node to the management server and stored in the /.ssh/known_hosts file on the management server.
6. The **updatenode** command calls the **/usr/bin/ssh** command to execute the /tmp/.ssh/copy.perl script on the node. The root password stored in the memory variable is used again. This script performs the following actions on the node:
   - Creates the /.ssh directory, if it does not exist.
   - Copies /tmp/authorized_keys and /tmp/authorized_keys2 to the /.ssh directory.
   - Deletes the /tmp/.ssh directory and its contents.

The ssh client and sshd daemon configuration files and are not touched during the **updatenode** command execution. The default values in these files are enough for the automated configuration. You can customize those files, but it is not required by default. Your customized configuration files can be distributed from the management server to your nodes using the **cfm** command.

The **cfm** command is described in *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859.

5.4.6 Verifying the SSH configuration

To verify proper ssh configuration, try to execute a **date** command on the node from the management server:

```
ssh -n node1 date
```

If this is successful, try **dsh**:

```
dsh -n node1 date
```

Test all nodes by issuing:

```
dsh -N AllNodes date
```

If it does not work, try the following:

- Check the files in the /.ssh directory on both the management server and the nodes.
- Check whether the sshd daemon is running on the node.
- Try to restart the sshd daemon on the node and check the output for possible errors or check the logs.
- Finally, retry the **updatenode** command on the management server:

  ```
  updatenode -k -n <node_name>
  ```

5.5 Other remote command execution programs

If you want to use other remote command execution software than OpenSSH, you need to configure it manually.

First, you should install and configure your remote shell to be able to execute any command on the node from the management server. Then verify that it works.

In the following example, we have a remote shell called my_remote_shell and a node called node1:

```
my_remote_shell -n node1 date
```

Finally, if you want this program to be used by default in your CSM cluster, update the CSM database RemoteShell attribute and SetupRemoteShell attribute. The RemoteShell attribute should be set to the fullname of your program. The SetupRemoteShell attribute should be set to 0 to prevent attempts to configure it automatically by the **updatenode** command.

Here is an example of how to set the default remote shell to the /usr/bin/my_remote_shell program:

```
csmconfig -r /usr/bin/my_remote_shell SetupRemoteShell=0
```

If you want this remote shell to be used only temporarily by the **dsh** command, you have to set the environment variable DSH_REMOTE_CMD:

```
export DSH_REMOTE_CMD=/usr/bin/my_remote_shell
```

If you want to use this remote shell only for individual **dsh** commands use the **-r** flag of the **dsh** command:

```
dsh -r /usr/bin/my_remote_shell -n node1 date
```

6

Security administration

This chapter explains some of the security administration tasks that an administrator may carry out. The discussion assumes that the reader is familiar with the CSM security architecture and components.

We cover the following topics:

- Administration of Cluster Security Services
- Administration of Resource Monitoring and Control

6.1 Administration of Cluster Security Services

By default, Cluster Security Services (CtSec) does not need to be configured in a CSM environment. All necessary configuration is done by CSM. This section explains the configuration files of CtSec and how to configure them if you choose to do so.

6.1.1 Configuration files

By default, CtSec does not modify any of the RSCT configuration files. CtSec is part of RSCT. The RSCT configuration files are located in /usr/sbin/rsct/cfg. These files should not be changed. If an administrator needs to modify one of these files, create a copy of it in /var/ct/cfg and modify the copy. RSCT searches /var/ct/cfg for configuration files first. If a file exists in this directory, it uses it. If it does not exist, then RSCT uses the original configuration file found in /usr/sbin/rsct/cfg.

Important: You should not modify any of the original RSCT configuration files in /usr/sbin/rsct/cfg. To modify a file, first copy it to /var/ct/cfg, and then modify the copy in that directory only.

Table 6-1 shows some of the important CtSec configuration files and when you may need to modify them.

Table 6-1 Configuration files for CtSec

Configuration file name	When to change?
ctsec.cfg	Configuration of MPM priorities, see 6.1.2, "Mechanism pluggable module configuration" on page 82
ctcasd.cfg	Configuration of ctcasd and its key generation, see 6.1, "Administration of Cluster Security Services" on page 82
ctsec_map.global	Administration of local mapped identities, see 3.2.5, "Identity mapping service" on page 29

6.1.2 Mechanism pluggable module configuration

By default, CtSec does not change mechanism pluggable module (MPM) priorities. For this reason, there is no copy of ctsec.cfg in /var/ct/cfg. Example 6-1 on page 83 shows the original content of that configuration file.

Example 6-1 Initial contents of the ctsec.cfg file

```
cat /usr/sbin/rsct/cfg/ctsec.cfg
#Prior  Mnemonic     Code       Path                 Flags
#-------------------------------------------------------------------------------
 1      unix         0x00001    /usr/lib/unix.mpm    i
```

As explained in 3.2.2, "Mechanism pluggable module (MPM)" on page 27, there is only one MPM shipped and configured for CtSec, so it makes no sense to change the priority of that MPM. However, you can turn off authentication by commenting out the line that defines the UNIX MPM. After that, all requests within the cluster have an unauthenticated state. This usually causes all attempts to access resources to fail, unless RMC ACL entries have also been changed to allow unauthorized access. By disabling the UNIX MPM, you effectively disable cluster security. If you think you need to do this, see *IBM Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference*, SA22-7889 and contact IBM support for further assistance.

Important: We recommend that you do not to change the ctsec.cfg file, because there is no other MPM you can configure in the current release.

6.1.3 The ctcasd daemon administration

The UNIX MPM uses ctcasd for host-based authentication, as described in 3.2.4, "Host-based authentication with ctcasd" on page 28.

The ctcasd daemon starts automatically when AIX 5L has been installed. You can check whether the ctcas subsystem is active with the following command:

```
lssrc -s ctcas
```

The output should be similar to the following:

```
Subsystem         Group            PID         Status
 ctcas            rsct             14562       active
```

To check whether the daemon is running, issue **`ps -efa|grep ctcasd`**.

The output should be similar to this:

```
root 14562  8608   0 18:30:52      -  0:05 /usr/sbin/rsct/bin/ctcasd
```

If you change the ctcasd configuration file, you must restart ctcasd with:

```
stopsrc -s ctcas
startsrc -s ctcas
```

6.1.4 The ctcasd daemon key files

When the ctcasd daemon is started for the first time, it creates a pair of public and private keys for its host. Additionally, ctcasd creates a trusted host list (THL) file to store host names and linked public keys of other hosts. Initially, the file contains the server's own public host name and key.

Table 6-2 shows the location and permissions of the key files and the THL file.

Table 6-2 The ctcasd daemon key files and trusted host list file

ctcasd file	Location	Permissions (root.system)
Private key	/var/ct/cfg/ct_has.qkf	`-r--------`
Public key	/var/ct/cfg/ct_has.pkf	`-r--r--r--`
Trusted host list	/var/ct/cfg/ct_has.thl	`-r--r--r--`

To list all the hosts and keys in the THL file, issue the following command:

```
/usr/sbin/rsct/bin/ctsthl -l
```

Example 6-2 shows the output of this command. It is useful to check if a public key of the target node is in place if you encounter any problems within your cluster.

Example 6-2 Contents of the THL file

```
ctsthl: Contents of trusted host list file:
--------------------
Host name: lpar1
Identifier Generation Method: rsa512
Identifier Value:
120200a972baeb464b656a929bc639d36aab4be46634b99af00249afea57d4cfcc53767d1d794ce91ab444917ae079c
eccbfa66a21d68cccc2a0255b6f8f97658ca69b0103
--------------------
Host name: manserver1
Identifier Generation Method: rsa512
Identifier Value:
120200c9f3901b54c75b78681ed24b733d41a95f706f7c971e757a60190bf9f1796a717f042edf73e1d170958aa953f
d8985161698e675347560ea9d0e02198ea0bbf70103
--------------------
```

6.1.5 Generate new keys

In the current version of CSM and RSCT, the host keys generated by ctcasd do not expire and will not be changed automatically. However, you can change the keys on your nodes and the management server manually.

You must stop the RSCT subsystem that CSM uses before changing keys; otherwise, unpredictable results can occur. Stopping RSCT is impossible if nodes are part of another RSCT peer domain cluster (HACMP, for example), because this will cause that cluster to fail. Therefore, plan downtime for your cluster or affected nodes, or both, if you intend to change keys.

Example 6-3 shows the list of RSCT components used by CSM. These must all be stopped.

Example 6-3 RSCT components used by CSM

```
mgmtserver2#lssrc -g rsct
Subsystem         Group            PID          Status
 ctcas            rsct             73830        active
 ctrmc            rsct             368710       active
mgmtserver2#lssrc -g rsct_rm
Subsystem         Group            PID          Status
 IBM.ServiceRM    rsct_rm          307230       active
 IBM.ERRM         rsct_rm          290948       active
 IBM.AuditRM      rsct_rm          200812       active
 IBM.HostRM       rsct_rm          233664       active
 IBM.CSMAgentRM   rsct_rm          315496       active
 IBM.DMSRM        rsct_rm          323656       active
 IBM.FSRM         rsct_rm          458766       active
```

Changing keys on nodes

The following steps describe how to change keys on nodes:

1. Stop RSCT on the management server and the nodes where you want to change the keys:

   ```
   stopsrc -g rsct
   stopsrc -g rsct_rm
   ```

   Check that all subsystems are inoperative on the management server and the nodes:

   `lssrc -g rsct` and `lssrc -g rsct_rm`

2. Delete the key files on the nodes:

   ```
   rm /var/ct/cfg/ct_has.qkf
   rm /var/ct/cfg/ct_has.pkf
   rm /var/ct/cfg/ct_has.thl
   ```

3. Delete the node's host public key from the THL file on the management server:

   ```
   /usr/sbin/rsct/bin/ctsthl -d -n nodes-hostname
   ```

   Check that they have been deleted successfully:

   ```
   /usr/sbin/rsct/bin/ctsthl -l
   ```

4. Start RSCT on the management server and all nodes:

   ```
   startsrc -g rsct
   ```

   Check if all subsystems have been started:

   `lssrc -g rsct` and `lssrc -g rsct_rm`

   If one of subsystems is inoperative, try to start them manually:

   ```
   startsrc -s subsystem_name
   ```

5. Run the **`updatenode -k`** command on the management server for all nodes where keys have been changed.
6. Verify the exchanged public keys as described in 6.1.8, "Verifying exchanged public host keys" on page 89.

Changing keys on the CSM management server

Important: If you want to change the keys on the management server, you need to follow the instructions for *all* nodes.

The following steps describe how to change keys on the management server:

1. Stop RSCT on management server and all nodes within the cluster:

   ```
   stopsrc -g rsct
   stopsrc -g rsct_rm
   ```

   Check that all subsystems are inoperative on the management server and all nodes:

   `lssrc -g rsct` and `lssrc -g rsct_rm`

2. Delete the key files on the management server:

   ```
   rm /var/ct/cfg/ct_has.qkf
   rm /var/ct/cfg/ct_has.pkf
   rm /var/ct/cfg/ct_has.thl
   ```

3. Delete the management server's public host key from the THL file on all nodes:

   ```
   /usr/sbin/rsct/bin/ctsthl -d -n managementserver_hostname
   ```

   Check that they have been deleted successfully:

   ```
   /usr/sbin/rsct/bin/ctsthl -l
   ```

4. Start RSCT on the management server and all nodes:

   ```
   startsrc -g rsct
   ```

   Check if all subsystems have been started:

   `lssrc -g rsct` and `lssrc -g rsct_rm`

   If one of subsystems is inoperative, try to start them manually:

   ```
   startsrc -s subsystem_name
   ```

5. Run the **`updatenode -k`** command on the management server for all nodes.

Verify the exchanged public keys as described in 6.1.8, "Verifying exchanged public host keys" on page 89.

6.1.6 Changing the default key type for ctcasd

By default, ctcasd works out of the box and uses the RSA512 algorithm to generate 512-bit keys. This is defined in /usr/sbin/rsct/cfg/ctcasd.cfg. This type of key generation is at least as secure as Kerberos 5 security. However, if you want to change the method for key generation (for example, to use RSA1024), you can modify the ctcasd configuration file.

Changing the configuration to use another key generation method will take effect only on newly generated keys. The old keys will be in place as they were generated.

The following steps describe how to change the default key type:

1. Copy the original ctcasd configuration file from /usr/sbin/rsct/ctcasd.cfg to /var/ct/cfg:

   ```
   cp /usr/sbin/rsct/ctcasd.cfg /var/ct/cfg
   ```

   Edit the file and change the value for HBA_KEYGEN_METHOD.

   Appropriate values can be rsa512 or rsa1024. To get a list of supported key generation methods, issue **`ctskeygen -i`**.

2. Restart ctcasd:

   ```
   stopsrc -s ctcas
   startsrc -s ctcas
   ```

Note: The ctcasd configuration file contains more parameters than mentioned here, but only a few of them are used or can be modified (for example, HBA_USING_SSH_KEYS is not supported at this time).

For further information about the parameters in the ctcasd configuration file, see *IBM Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference*, SA22-7889.

6.1.7 Removing entries from the trusted host list file

If you remove a node from your CSM cluster, the **`rmnode`** command does not remove the host keys entries from the THL files, either on the CSM management server, or on the managed node. It is not a security problem, but it is not a clean removal of the node. If you want to remove the public host keys after removing a node from a cluster, you need to do the following steps.

On the CSM management server:

1. Run the **`/usr/sbin/rsct/bin/ctsthl -d -n node1`** command.
2. Check if key has been removed with **`/usr/sbin/rsct/bin/ctsthl -l`**.

On a node that has been removed:

1. Run the **`/usr/sbin/rsct/bin/ctsthl -d -n csmserver1`** command.
2. Check if key has been removed with **`/usr/sbin/rsct/bin/ctsthl -l`**.

Note: Be careful removing public host keys. Removing the wrong keys can cause cluster communication to fail.

6.1.8 Verifying exchanged public host keys

To guard your cluster against spoofing and attacks, you should verify the public host keys that your cluster nodes exchanged by issuing the **installnode** and **updatenode** commands. To do this, complete the following steps:

1. Log on to the node whose public key has been copied to the management server. You should log on either locally or using a secure shell only.

   Execute the following command to write a text version of the node's public key to a file:

   ```
   /usr/sbin/rsct/bin/ctskeygen -d >/tmp/hostname_pk
   ```

   Depending on the configured key generation method, the output should look like this:

   ```
   120200f6cf79d2438f3925175a9fccc8296528d2c6749d51186d09af72444a2770bd6193a11
   d603d7fb74c0579975cae859744d2e484ed573ba0a2673c2f48ded2b8f30103
   (generation method: rsa512)
   ```

2. Log on to the management server. You should log on either locally or using a secure shell only.

   Execute the **usr/sbin/rsct/bin/ctsthl -l** command to verify the key in the THL file. The output for the specified node should look similar to the output shown in Example 6-4.

Example 6-4 Output of the ctsthl -l command

```
Host name: node1
Identifier Generation Method: rsa512
Identifier Value:
120200f6cf79d2438f3925175a9fccc8296528d2c6749d51186d09af72444a2770bd6193a11d603
d7fb74c0579975cae859744d2e484ed573ba0a2673c2f48ded2b8f30103
```

If the keys are not the same, either the transfer of the keys contained failures, or someone or something is spoofing the identity of that node.

6.2 Administration of Resource Monitoring and Control

By default, Resource Monitoring and Control (RMC) does not need to be configured in a CSM management environment. However, there are some situations where you may need to perform some RMC administration tasks.

6.2.1 Configuration files for Resource Monitoring and Control

RMC is part of RSCT. The original RSCT configuration files are located in /usr/sbin/rsct/cfg. These original files should not be changed. If CSM needs to modify one of these files it creates a copy of it in /var/ct/cfg and modifies the copy. RSCT searches /var/ct/cfg for configuration files first. If the file exists in this directory, it uses it. If it does not exist, RSCT uses the original configuration file found in /usr/sbin/rsct/cfg.

Important: You should not modify any of the original RSCT configuration files in /usr/sbin/rsct/cfg. To modify a file, first copy it to /var/ct/cfg, and then modify the copy in that directory only

Table 6-3 shows an important configuration file for RMC, if it is copied from the original place to /var/ct/cfg, and when you need to modify it.

Table 6-3 Configuration file for Resource Monitoring and Control

Configuration file name	Copied to /var/ct/cfg	When to change?
ctrmc.acls	Yes	Administration of client access to resources

6.2.2 Allowing a non-root user to administer CSM

Adding non-root users to the RMC environment is necessary if you want to allow other users to administer or monitor the Cluster Systems Management (CSM) cluster without being root. In order to do this, you need to change to the following configuration files:

- ctrmc.acls
- ctsec_map.global, if local identity mapping will be used

If you want to add more than one user, or if the user administers the cluster from different hosts, it may be easier to use local identity mapping. To understand how mapping works, see 3.2.5, “Identity mapping service” on page 29.

The following two examples show how to configure RMC to allow non-root users access to cluster resources.

The examples allow the user read and write access to every resource. If you want to allow read-only access (for example, for monitoring only), you need to change the permissions. If, in either case, you do not want to allow access to all resources, you need to specify permissions for each resource. For more details about adding users and permissions to the ACL file, see *IBM Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference*, SA22-7889.

Allowing a user ID to administer the cluster from one server

This example adds one user to the configuration files. This user can administer the cluster from the management server only. The RMC ACL file /var/ct/cfg/ctrmc.acls on the management server and all the nodes this user can administer must be changed.

1. On the management server, open the file with **`vi /var/ct/cfg/ctrmc.acls`** and go to the DEFAULT stanza.
2. Edit the stanza to add user *david*, as shown in Example 6-5.

Example 6-5 Allowing a user ID to administer the cluster from one server: The ACL file

```
DEFAULT
    root@LOCALHOST      *       rw
    david@csmserver1    *       rw
    LOCALHOST           *       r
```

3. Save the file and distribute it to all the nodes using **`dsh`** or CFM.

Allowing a user ID to administer the cluster from any node

This example adds one user to the configuration files. This user can administrate the cluster from every node within the cluster. The RMC ACL file /var/ct/cfg/ctrmc.acls on the management server and all nodes this user can administer must be changed. For easier administration, this example also uses local identity mapping.

Add the local identity mapping entry to the mapping file:

1. Copy the global mapping file to /var/ct/cfg if the file does not exist. If the file already exists, do not overwrite it.

   ```
   cp /usr/sbin/rsct/cfg/ctsec_map.global /var/ct/cfg/
   ```

2. Edit this file with **`vi /var/ct/cfg/ctsec_map.global`** and add the following line to allow user david access to the cluster resources:

   ```
   unix:david@<cluster>=mapped_david
   ```

   This maps user david coming from every node within the active cluster to the local identity mapped_david. Remember, this mapped identity does not need to exist as a real user ID on the operating system.

   You also can map this user to the local identity root by adding the following line:

   ```
   unix:david@<cluster>=root
   ```

   In this case, you do not need to change the RMC ACL file, because root already has all permissions to access the resources within the cluster.

3. If you did not map the user to the local identity root, you need to change the ACL file. Open the file with **`vi /var/ct/cfg/ctrmc.acls`** and go to the DEFAULT stanza.
4. Edit the stanza to add user mapped_david, as shown in Example 6-6.

Example 6-6 Allowing a user ID to administer the cluster from any node: The ACL file

```
DEFAULT
    root@LOCALHOST      *       rw
    mapped_david        *       rw
    LOCALHOST           *       r
```

5. Distribute the global mapping file and the RMC ACL file to the management server and the nodes using **`dsh`** or CFM.

Abbreviations and acronyms

ACL	access control list
AIX	Advance Interactive Executive
API	application program interface
CA	certificate authority
CCDB	Control Context Data Buffer
CDMF	Commercial Data Masking Facility
CFM	configuration file manager
CSM	Cluster Systems Management
ctcasd	Cluster Technology Cluster Authentication Service daemon
CtSec	Cluster Security Services
DCEM	Distributed Command Execution Manager
DES	Data Encryption Standard
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
DSH	Distributed Shell
GID	group identification
GPFS	General Parallel File System
HACMP	High-Availability Cluster Multiprocessing
HBA	host-based authentication
HMC	Hardware Management Console
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IDEA	International Data Encryption Algorithm
IDM	Identity Mapping
IP	Internet Protocol
ITSO	International Technical Support Organization
MAL	mechanism abstract layer
MD	Message Digest
MPM	mechanism pluggable module
NFS	Network File System
NIM	Network Installation Manager
NIST	National Institute of Standards and Technology
NSA	National Security Administration
PSSP	Parallel System Support Program
RM	resource manager
RMC	Resource Monitoring and Control
RSA	Rivest, Shamir, and Adelman
RSA	Remote Supervisor Adapter
RSCT	Reliable Scalable Cluster Technology
SHA	Secure Hash Algorithm
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
THL	trusted host list
TLS	Transport Layer Security
UID	user identification
VLAN	virtual local area network
WebSM	Web-Based System Manager

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see "How to get IBM Redbooks" on page 96.

- *An Introduction to CSM 1.3 for AIX 5L,* SG24-6859

Other resources

These publications are also relevant as further information sources:

- *IBM Reliable Scalable Cluster Technology for AIX 5L: RSCT Guide and Reference*, SA22-7889
- *AIX 5L Version 5.2 Web-based System Manager Administration Guide*. It is available online at:

  `http://publib16.boulder.ibm.com/doc_link/en_US/a_doc_lib/aixbman/wsmadmn/wsmadmn.pdf`

Referenced Web sites

These Web sites are also relevant as further information sources:

- Conserver Web site

  `http://www.conserver.com`
- IBM AIX Toolbox for Linux Applications Web site

  `http://www.ibm.com/servers/aix/products/aixos/linux/download.html`
- OpenSSH Version 3.4 for AIX 5L Version 5.2 download

  `ftp://www-126.ibm.com/pub/opensshi/3.4p1_52/openssh34p1_52.tar.Z`
- Bull Freeware Web site

  `http://www.bullfreeware.com/`

- Cryptographic content for AIX Web site (requires registration)

  `http://www6.software.ibm.com/dl/aixtbx/aixtbx-h?S_PKG=dlaixww&S_TACT=&S_CMP=`
- Open Source Software for AIX Web site

  `http://oss.software.ibm.com/developerworks/downloads/?group_id=108`
- OpenSSL Software for AIX Download Web site

  `http://www6.software.ibm.com/dl/aixtbx/aixtbx-i?S_PKG=dlaixww&S_TACT=&S_CMP=`
- OpenSSH Version 3.4 Download Web site (Bull version)

  `http://www.bullfreeware.com/download/aix43/openssh-3.4.0.0.exe`

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.

Index

IBM Redbooks

An Introduction to Security in a CSM 1.3 for AIX 5L Environment

An Introduction to Security in a CSM 1.3 for AIX 5L Environment

Peek at the latest security mechanisms for pSeries clusters

Practical security considerations included

Security concepts and components explained

This IBM Redbook contains information about the first official release of the new clustering software IBM Cluster Systems Management (CSM) on AIX 5L Version 5.2. Features include base cluster configuration and management, Resource Monitoring and Control (RMC), subsystem access control list setup for shipped CSM resource managers, hardware control, configuration file management, distributed command execution, and a distributed GUI based on the AIX WebSM infrastructure. Included in this release of CSM is a complete set of base security functions based on IBM host-based authentication (HBA) and offered through an abstraction layer in the CSM software. CSM automatically configures HBA for use by the cluster services and establishes secure cluster communications for the shipped CSM resource managers.

The first part of this publication is conceptual and includes an introduction to security for CSM 1.3 for AIX 5L, security concepts and components, and CSM security infrastructure. Next, practical security considerations are provided. Topics, such as network considerations, security in an heterogeneous environment, and security considerations for hardware control, are discussed. The last part of this publication details secure remote command execution, as well as security administration. Among the topics covered are remote command execution software, OpenSSH installation, and administration of RMC.

SG24-6873-00 ISBN 0738427896