SSH Tectia® Client/Server 6.1

Product Description

30 November 2009

SSH Tectia® Client/Server 6.1: Product Description

30 November 2009

For Open Source Software acknowledgements, see appendix *Open Source Software License Acknowledgements* in the *Product Description*.

SSH Communications Security Corp.
Kumpulantie 3, FI-00520 Helsinki, Finland

Table of Contents

Chapter 1 About This Document

This document describes the SSH Tectia client/server solution and its components. This document is intended as background information for persons responsible for the deployment and operation of the solution.

The SSH Tectia client/server solution consists of the following products that can be purchased and installed separately, but ideally, the SSH Tectia client- and server-side components work together:

- SSH Tectia Client
- SSH Tectia ConnectSecure
- SSH Tectia Server
- SSH Tectia Server for Linux on IBM System z

This document contains the following information:

- Introduction to the SSH Tectia client/server solution
- Features and benefits provided to the administrators and users
- Architecture of the components, their configuration methods and the authentication methods used
- Use cases which show how SSH Tectia client/server solution can be applied in enterprises
- Product specification gives technical data
- Glossary explains the basic Secure Shell -related terms and concepts

For instructions on installing and using the SSH Tectia products, refer to the product-specific manuals:

- *SSH Tectia Client/Server Quick Start Guide* gives instructions on how to get started with SSH Tectia Client and Server, for example for evaluation purposes. There are two guides, one for Windows, and the other for Unix and Linux environments.
- *SSH Tectia Client User Manual* gives instructions for installing and using SSH Tectia Client on Unix and Windows platforms, and on Linux running on IBM System z.

- *SSH Tectia ConnectSecure Administrator Manual* contains instructions for installing and using SSH Tectia ConnectSecure on Unix and Windows platforms.

- *SSH Tectia Server Administrator Manual* contains instructions for installing and using SSH Tectia Server on Unix and Windows platforms, and on Linux running on IBM System z.

In addition to the SSH Tectia Client Server and ConnectSecure, SSH offers efficient tools for handling MFT environments. For information on those tools, refer to their own product descriptions available at:

http://www.ssh.com/support/.

Information on other SSH Tectia products is available in their own documents:

- *SSH Tectia MFT Auditor Product Description* contains information on the architecture and the features of SSH Tectia MFT Auditor that is a separate auditing and monitoring tool for MFT environments.

- *SSH Tectia MFT Events Product Description* contains information on the architecture and the features of SSH Tectia MFT Events that is a separate product for managing file transfer events in enterprise MFT environments.

- *SSH Tectia Manager Product Description* contains information on the architecture and the features of SSH Tectia Manager that is a centralized tool for managing the SSH Tectia product installations and configurations in enterprise environment.

Chapter 2 Introduction

The SSH Tectia client/server solution is designed to secure end-to-end communications within corporate networks. SSH Tectia allows secure network services over an unsecured network, such as the Internet. SSH Tectia products can be deployed cost-effectively also to very large corporate networks, and the installation and maintenance can be managed centrally.

SSH Tectia products are based on Secure Shell (SecSh) technology originally developed by the founders of SSH Communications Security. The Internet Engineering Task Force (IETF) has standardized the Secure Shell protocol, see RFC 4251 at http://www.ietf.org/rfc/rfc4251.txt.

The award-winning SSH Secure Shell technology allows you to securely log in to remote host computers, to execute commands safely on a remote computer, to securely copy remote files, to forward X11 sessions (on Unix), and to provide secure encrypted and authenticated communications between two non-trusted hosts. Arbitrary TCP/IP ports can also be forwarded (tunneled) over the secure channel, enabling secure connection, for example, to an e-mail service.

The SSH Tectia client/server solution includes the following separate products:

- SSH Tectia Client - a workstation product providing the basic Secure Shell client features and tools
- SSH Tectia ConnectSecure - installed on server hosts for securing server-to-server connections, while providing enterprise-level secure file transfer features in addition to the basic Secure Shell client features
- SSH Tectia Server - the Secure Shell server product
- SSH Tectia Server for Linux on IBM System z - Secure Shell server installed on Linux on IBM System z platforms. It provides both the client-side and server-side Secure Shell components. The client-side components are identical with SSH Tectia Client, so in this document, what is said about SSH Tectia Client applies also to the client-side components.
- SSH Tectia Server for IBM z/OS - Secure Shell server installed on IBM mainframes. It provides both the client-side and server-side Secure Shell components. The client-side components are identical with SSH Tectia Client, so in this document, what is said about SSH Tectia Client applies also to the client-side components. SSH Tectia Server for IBM z/OS is introduced in its own *Product Description*.

The SSH Tectia products work ideally together, but they can also be used with other Secure Shell-based clients or servers.

SSH Tectia solution suite offers a versatile enterprise platform for governance of system security and data-in-transit, automation of file transfers and business processes, as well as managed file transfer and security infrastructure as shown in Figure 2.1.

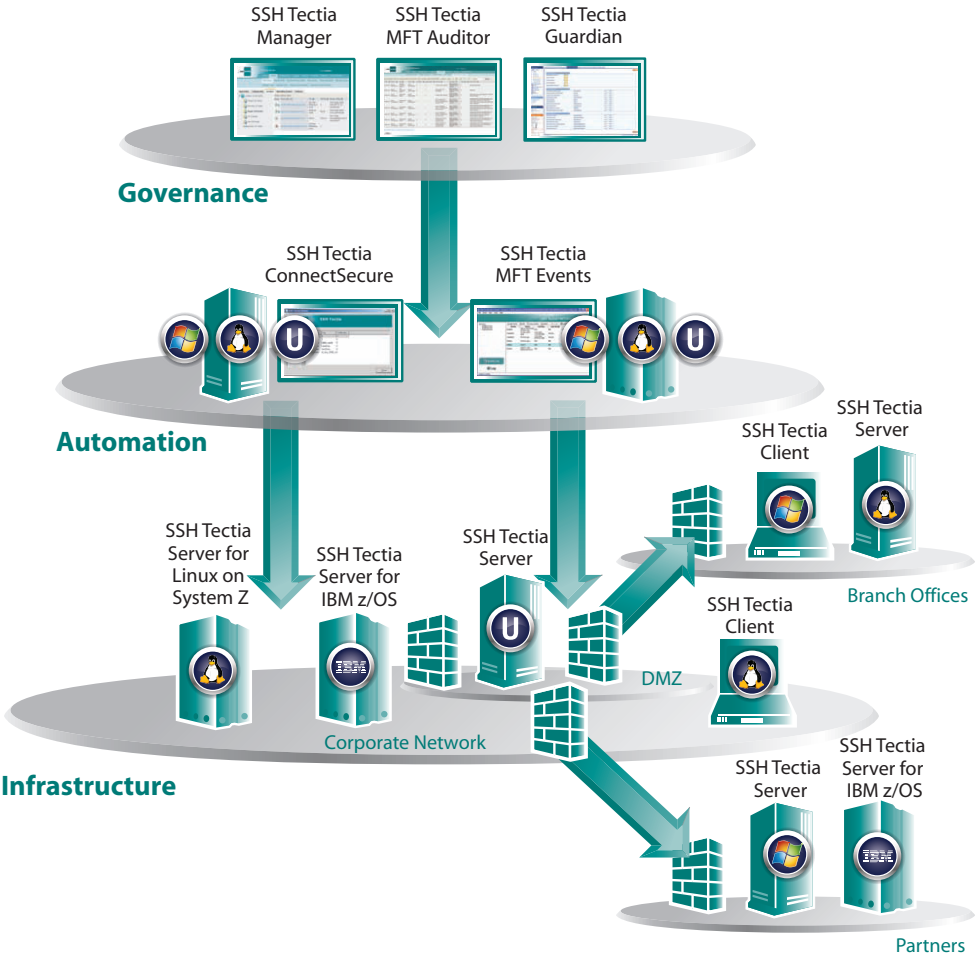

Figure 2.1. SSH Tectia product offering

SSH Tectia Client and Server are part of the infrastructure layer, providing a powerful and versatile framework for managed file transfer, application connectivity, and secure system administration. SSH Tectia ConnectSecure operates on the automation layer, providing tools for composing, automating, and executing secure file transfer operations and other data transactions without expensive re-engineering of applications and scripts.

2.1 Key Applications

The key applications of SSH Tectia client/server solution are secure file transfer, secure application connectivity, and secure system administration.

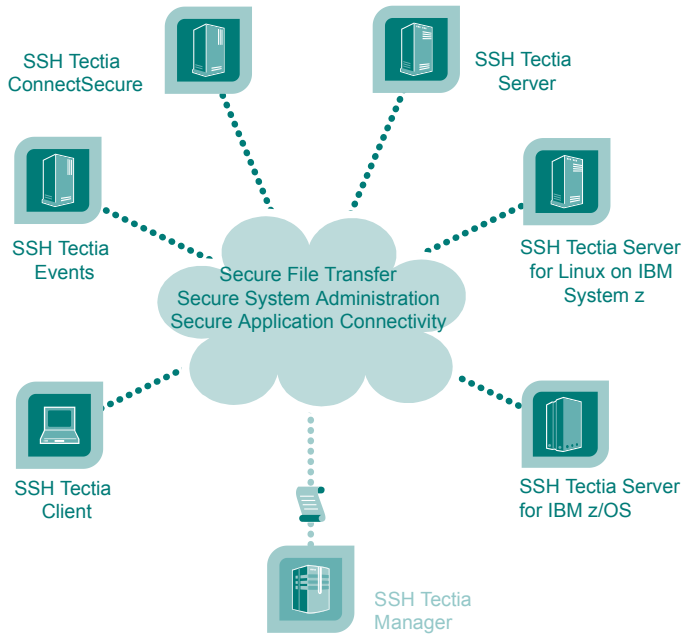

Figure 2.2. The key applications of SSH Tectia products

2.1.1 Secure File Transfer

The SSH Tectia client/server solution allows organizations to replace plaintext file transfer protocol (FTP) connections with secure file transfers in cross-platform environments. File transfers can be secured by applying the Secure File Transfer Protocol (SFTP) instead of FTP, or by using tunnels that encrypt the connection from the FTP client to the FTP server.

The third-generation high-performance Secure Shell protocol implementation, SSH G3, provides unparalleled SFTP throughput and scalability, eliminating processing bottlenecks and helping to meet critical deadlines.

SSH Tectia Server is capable of handling also the OpenSSH SCP legacy file transfer mechanism that OpenSSH uses instead of the standard SFTP protocol. The OpenSSH SCP can interoperate with the SSH Tectia Server running on any platform.

The SSH Tectia client/server solution offers three methods for FTP replacement as illustrated in Figure 2.3:

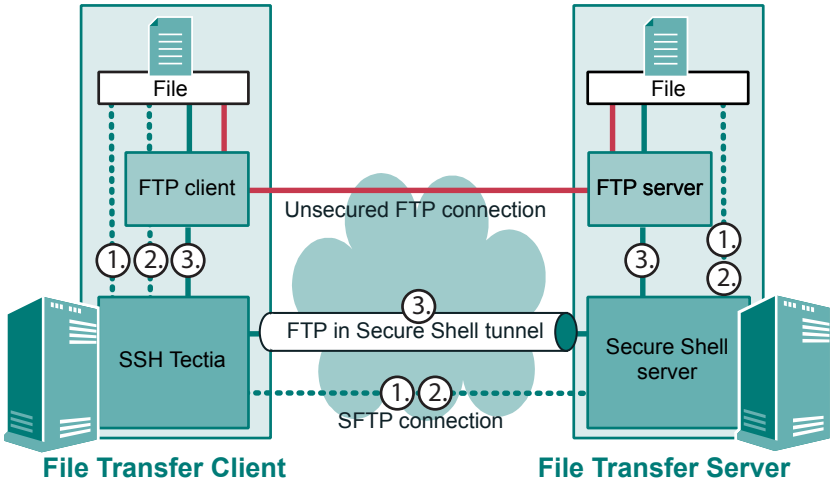

Figure 2.3. Options for replacing unsecured FTP file transfers

An **unsecured FTP connection** is shown in red. If this is used, user IDs, passwords and the actual transferred data are sent in plaintext, which makes them vulnerable to eavesdropping and unathrorized modifications.

SSH Tectia products use the following methods to make file transfers secure:

1. **Native SFTP**

   The secure file transfer protocol (SFTP) transfers the files and the related control data in encrypted format. SFTP can be activated by using the `sftpg3` and `scpg3` tools, or the SSH Tectia file transfer GUI (on Windows) instead of the unsecured `ftp` tools.

   SSH Tectia Client or ConnectSecure provides the SFTP functionality and connects to any Secure Shell SFTP server. Both the original FTP client and FTP server can be eliminated.

2. **FTP-SFTP conversion**

   Connections from the original FTP client are transparently captured by SSH Tectia ConnectSecure, converted to SFTP, and directed to a Secure Shell SFTP server. No changes to the original FTP client application are needed, and it can remain being used as before. The original FTP server, however, is eliminated.

   This feature is available with SSH Tectia ConnectSecure and SSH Tectia Server for IBM z/OS (client tools) on all supported platforms and requires a Secure Shell server as the counterpart.

   For more information, see Section 3.1.2.

3. **Transparent FTP tunneling**

   Transparent FTP tunneling creates a secure tunnel between an FTP client and an FTP server. All material is sent in encrypted format and so secured from eavesdropping. This feature is available with SSH Tectia ConnectSecure and SSH Tectia Server for IBM z/OS (client tools).

For more information, see Section 3.1.3.

The SSH Tectia client/server solution supports also non-transparent FTP tunneling on both SSH Tectia Client and ConnectSecure. Non-transparent FTP tunneling can be implemented as SOCKS tunnels defined in the SSH Tectia connection profiles, or as automatic tunnels defined in the Connection Broker configuration.

2.1.2 Secure Application Connectivity

The SSH Tectia client/server solution can be used to replace unsecured TCP-based terminal connections (for example Telnet) to business-critical enterprise applications. Through strong encryption and data integrity, the SSH Tectia client/server solution protects sensitive data and passwords against unauthorized access, facilitating compliance with regulations and best practices. Strong authentication of users is supported through broad integration with third-party authentication systems, including RSA SecurID and public-key infrastructure (PKI).

With the transparent TCP tunneling feature activated on the client-side (on SSH Tectia Client or ConnectSecure), TCP-based applications can be tunneled transparently without changing the end-user experience or requiring any modifications on the applications, thus reducing the total cost of ownership.

SSH Tectia Client and ConnectSecure can also be used for application protection using the static tunneling feature. As opposed to transparent TCP tunneling, static tunnels are configured so that an application connects to a local port running SSH Tectia Client, and the Client tunnels the application to a specified remote host.

For more information, see Section 3.3.

2.1.3 Secure System Administration

The SSH Tectia client/server solution is used by system administrators as a replacement for the unsecured login protocols, Rlogin, Telnet, and FTP. With the SSH Tectia client/server solution, the administrators can login to remote hosts securely, as their user ID and authentication information are transmitted in encrypted format over the network.

The SSH Tectia Client software is installed in the system administrator's workstation and the SSH Tectia Server software in the managed server. In such a configuration, the number of servers can be much higher than the number of client installations. In numerous active SSH Tectia implementations there are thousands of SSH Tectia Server instances installed within a corporate network.

With the SSH Tectia client/server solution, login can be easily done also in heterogeneous network environments including Windows, Unix, Linux, and IBM mainframe systems. This eliminates the need to deploy and maintain Secure Shell implementations from multiple vendors.

2.2 Origins of Secure Shell

The Secure Shell concept originated on Unix as a replacement of the unsecured "Berkeley services", that is, the `rsh`, `rcp`, and `rlogin` commands. Secure Shell replaces also other unsecured terminal applications such as Telnet and FTP.

An increasing number of remote access tasks involve the exchange of confidential data over unsecured TCP/IP networks. A typical example of a remote access task is business e-mail where confidentiality of the data and authentication of the user are highly desired.

The core communication protocols used on the Internet do not natively provide confidentiality for data. Security services are thus deployed to protect the transmitted data from monitoring and modification by unauthorized parties. Security services eliminate many threats that exist on the Internet.

Passive Attacks

In a passive attack, the attacker monitors and maybe records the data that passes by on the network. Examples of passive attacks are eavesdropping and traffic analysis. Passive attacks are very hard to detect since they leave little or no trace of activity.

Active Attacks

In active attacks, the attacker takes an active part in the communication. The attacker modifies or deletes data belonging to the stream coming from a legitimate party, inserts extra data to the stream, or initiates direct connections. Examples of active attacks are IP spoofing, TCP hijacking, replay, routing spoofing, and denial of service (DoS). Active attacks are usually easier to detect, but they also cause most harm.

2.2.1 Security Services

Secure remote access technology essentially requires three core security services:

- **Confidentiality:** The transmitted data must not be readable by unauthorized parties on the network. Confidentiality is achieved through encryption.
- **Integrity:** Unauthorized parties must not be able to modify the data without detection. Integrity is achieved by using checksum values, which reveal tampering attempts at the receiving end.
- **Authentication:** Both communicating parties must be able to identify each other reliably, so that no one else can pretend to be the other party. Authentication can be implemented by using challenge passwords, for instance. However, stronger authentication is achieved through public-key cryptography and digital signatures.

Non-repudiation is also usually mentioned along with these three services. Non-repudiation is a security service that prevents an entity from denying previous commitments or actions. However, in the context of communications security, non-repudiation is difficult to apply.

Note that the terms *authentication* and *authorization* refer to different actions. Authentication is the act of verifying the identity of an entity, whereas authorization is the act of verifying whether the identified entity is allowed to perform a task such as reading a file. Authentication usually precedes authorization. Authorization is determined by an access control system.

The Secure Shell protocol provides the confidentiality, integrity, and authentication services.

2.2.2 Secure Shell Protocol

Secure Shell secures connections over TCP/IP networks by encrypting passwords and other data. Once launched, it transparently provides strong authentication and secure communications over unsecured networks.

Secure Shell defines a packet-based protocol that runs over a reliable transport stream, usually TCP. The protocol does not run over UDP.

Secure Shell provides security at the application layer of the TCP/IP protocol stack. It is an application suite for providing secure access for diverse tasks in a flexible way, a versatile security solution that has become an essential tool in remote administration.

Secure Shell was originally developed to solve the two most acute problems in the Internet, secure remote terminal logins and secure file transfers. FTP and Telnet offer no protection for data and are easy targets for eavesdropping attacks. The primary goal of Secure Shell has been to replace these unsecured protocols with a secure one.

Secure Shell can also tunnel arbitrary TCP traffic over a single encrypted connection. Tunneling is a powerful feature that makes it possible to secure the communication of other applications and protocols without modifying the application code. By using tunnels, users can continue to use existing unsecured applications, such as e-mail and X11 applications, in a secure manner. With tunneling, Secure Shell can offer an encompassing solution for securing most of the communication tasks.

2.2.3 Different Protocol Versions

There are two versions of the Secure Shell protocol. The current version, Secure Shell version 2 (SecSh v2, SSH2) provides several security improvements as compared to the original Secure Shell version 1 (SecSh v1, SSH1). SSH Tectia is based on SecSh v2, as SSH Communications Security considers SecSh v1 deprecated and does not recommend nor support its use anymore.

SSH G3, based on and compatible with the standard SSH2, considerably improves Secure Shell encryption and management performance, securing throughput-intensive file transfers and applications without causing processing bottlenecks. The SSH G3 architecture is incorporated in the SSH Tectia client/server solution version 5.0 and later.

More information on the protocol can be found in the *Secure Shell RFCs (4250-4256)* (at http://www.ietf.org/rfc.html).

Chapter 3 Features and Benefits

The SSH Tectia client/server solution is an ideal light-weight solution for secure file transfer, secure remote logins for system administration purposes, and secure use of e-mail and other business applications. The SSH Tectia client/server solution is available for most Unix and Linux platforms (including Linux on IBM System z), for Microsoft Windows, and for IBM mainframes.

There are some differences in the availability of features between versions of the SSH Tectia products. For information on the features supported on each product version, see Table 6.3

3.1 FTP Replacement

The SSH Tectia client/server solution provides several methods for getting rid of the risks of plaintext File Transfer Protocol (FTP). SSH Tectia Server, Client and ConnectSecure all include a secure alternative, Secure File Transfer Protocol (SFTP) that can be used to replace existing FTP clients and servers. If replacing all FTP clients and servers in an environment is unfeasible, SSH Tectia ConnectSecure and SSH Tectia Server for IBM z/OS offer also transparent FTP tunneling to encrypt the FTP connection, and FTP-SFTP conversion to convert unsecured FTP traffic to use the secure SFTP protocol, instead.

SSH Tectia ConnectSecure can be installed on server hosts to secure server-to-server connections, and it can connect to any standard Secure Shell server. In addition to FTP replacement services, SSH Tectia ConnectSecure also provides enhanced file transfer (EFT) features.

The following table lists the benefits offered by each of the FTP replacement methods.

Table 3.1. Differences between FTP and secure file transfer methods

Feature	FTP	SFTP	FTP-SFTP Conversion	Transparent FTP Tunneling
Automated scripts can be used	x	x	x	x
FTP application used unmodified	x		x	x
Original FTP client running	x		x	x
Original FTP server running	x			x
Configured on SSH Tectia		x	x	x
Fallback to FTP is possible	N/A		x	x
SFTP GUI available on Windows	N/A	x	N/A	N/A

3.1.1 Secure File Transfer Protocol (SFTP)

Secure file transfer provides a secure, drop-in replacement for plaintext FTP. The Secure File Transfer Protocol functionality of the SSH Tectia client/server solution allows secure copying, moving, editing, and removing of files over TCP/IP networks.

The Secure File Transfer Protocol (SFTP) is a de-facto industry standard for secure file transfers, and it is natively supported by the SSH Tectia client/server solution.

Scripted file transfers

The SSH Tectia products include versatile command-line SFTP and SCP (Secure Copy) tools for easy scripting of automated and ad-hoc file transfers between enterprise servers.

File Transfer GUI on Windows

Easy-to-use graphical user interface for Windows allows users to securely drag-and-drop files between local Windows and remote Unix, Linux, Windows, and mainframe systems.

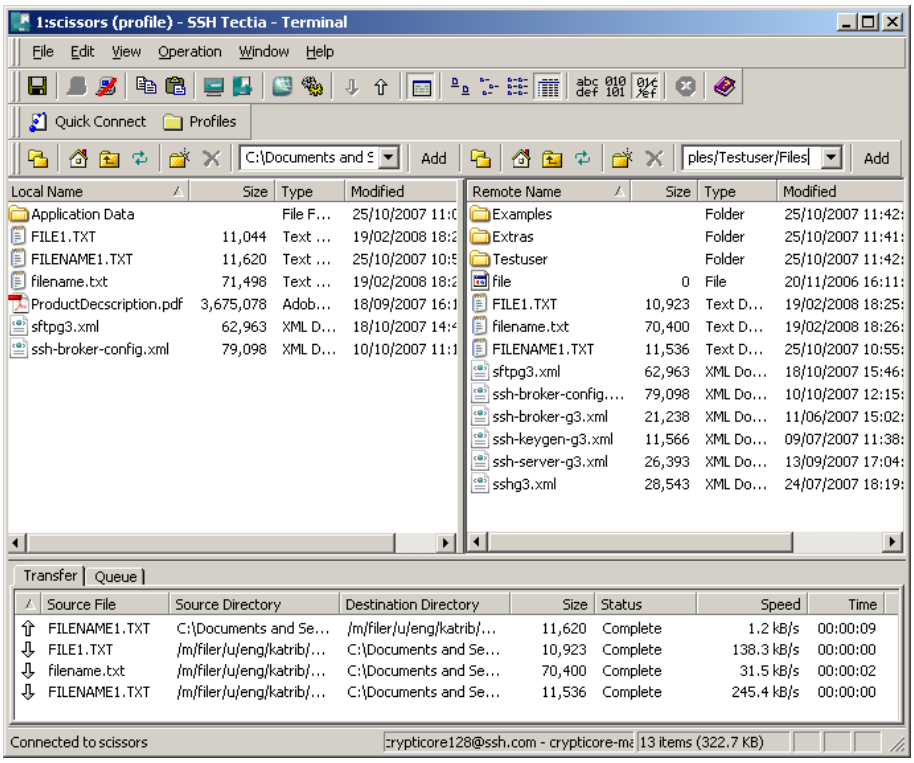

Figure 3.1. SSH Tectia File Transfer GUI

Migration to SFTP can be done gradually

Migration to secure file transfer can be made gradually. The SSH Tectia products can be configured to allow fallback to plaintext FTP, in case an SFTP connection cannot be established. This makes it possible to use SFTP where it is applicable, but still permits connections to those FTP servers that must remain operational, for example on a third-party network.

File transfer resume

The file transfer resume feature allows resuming interrupted file transfers instead of restarting the whole operation. The file transfer resume uses file hashing to determine the point of resume. For increased performance, you can apply the SSH Tectia ConnectSecure product together with SSH Tectia Server to activate a checkpoint/restart mechanism for optimum handling of interruptions in large file transfers.

Anonymous secure file transfers

SSH Tectia Server can be configured to allow anonymous file transfers in environments, where user authentication is not required. When anonymous authentication is in use, users do not need to type in a password.

Easy SFTP subsystem chrooting

SSH Tectia Server can be easily configured to confine users to a specific directory tree (e.g. home directory) for added security and ease of use.

CryptiCore® support

CryptiCore is applied to encryption and data integrity verification. The SSH G3 architecture and the high-speed CryptiCore algorithms (Intel platforms) help in meeting the performance requirements of large-scale application access scenarios. SSH Tectia ConnectSecure enables up to 400 Mb/s secure file transfer throughput in 1Gb networks and the destination server can be any Secure Shell server.

OpenSSH SCP support

SSH Tectia supports the legacy OpenSSH SCP implementation for easy migration of OpenSSH environments to SSH Tectia, creating a smoother transition to ensure seamless connectivity during the migration period.

3.1.2 FTP-SFTP Conversion

SSH Tectia ConnectSecure provides a FTP-SFTP conversion feature which captures plaintext FTP connections initiated by an FTP client and converts them to SFTP before the file transfer is started. All user names, passwords, and data are then transferred in encrypted format.

FTP-SFTP conversion works transparently

Existing FTP connections, including automated file transfers, can be transparently converted to SFTP without the need to modify the existing scripts or applications. Users can keep working with their familiar applications and use the existing IDs and authentication methods.

Easy and cost-effective conversion

The FTP-SFTP conversion module allows easy and cost-effective replacement of plaintext file transfers in large enterprise environments. Existing FTP scripts and client applications need no modifications. Only the FTP server will be replaced with an SFTP server.

Existing FTP clients can be used

Any existing client with FTP functionality can be used as before:

- Application hard-coded FTP
- Script-based automated FTP
- Interactive passive or active FTP, for example Windows Explorer FTP, web-browser-based FTP, command-line `ftp`, or FTP GUI applications.

Any Secure Shell server as counterpart

With SSH Tectia ConnectSecure, the FTP-SFTP conversion feature can connect to SSH Tectia Server or any other Secure Shell server. When SSH Tectia is used as the server-side conterpart, it can run on any supported platform: on Linux, HP-UX, AIX, Solaris, Windows, or IBM mainframe.

Easy filter rule configuration

SSH Tectia ConnectSecure can be configured to extract the user name, password, and destination host name from the secured FTP application, and to use them for authentication and connection setup on the Secure Shell SFTP server. The configuration is made as a filter rule in the Connection Broker configuration

file, and the same rule can be defined to cover all FTP traffic. In large FTP environments, this simple rule setting can save the effort of defining hundreds of connection profiles which would otherwise be needed separately for each destination.

The principle of FTP-SFTP conversion is shown in Figure 3.2.

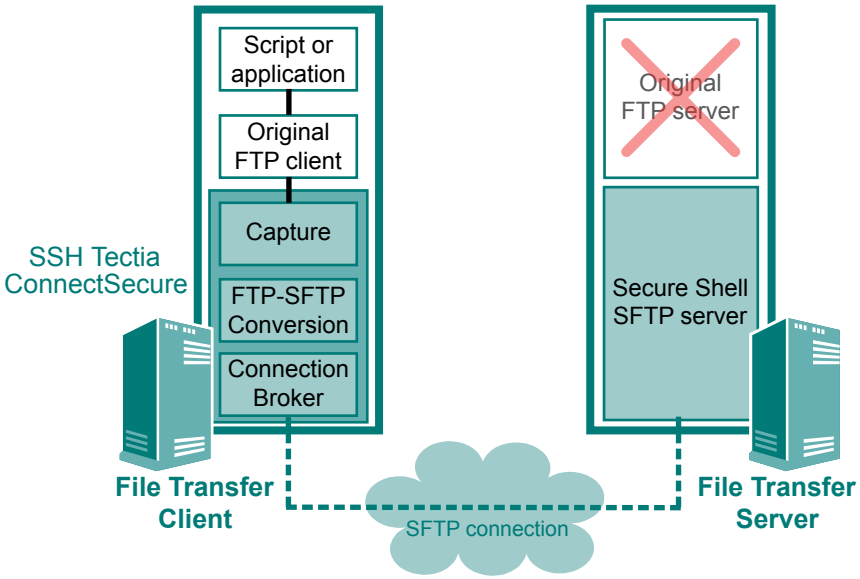

Figure 3.2. The architecture of FTP-SFTP conversion

The following steps happen during the FTP-SFTP conversion:

1. An application, a script, or a user triggers a file transfer.
2. The original FTP client in the File Transfer Client host starts opening a file transfer connection to the original destination FTP server (in File Transfer Server).
3. The SSH Tectia connection capture module captures the connection before it leaves the client side. SSH Tectia ConnectSecure checks and applies the filter rules that specify which connections to capture. The filter rules are defined in the Connection Broker configuration. Connections can be captured based on the FTP application used and the destination address and/or the port.
4. SSH Tectia ConnectSecure can extract the user name, password, and the destination host name from the secured FTP application, and use them for authentication and connection setup with the Secure Shell SFTP server.
5. The FTP-SFTP conversion module manages the FTP connection so that it remains unchanged from the original FTP client's point of view. FTP is converted to secure SFTP file transfer.
6. The SFTP connection is managed by the Connection Broker module.
7. The Secure Shell SFTP server in the File Transfer Server host is the end point of the file transfer.

The unsecured original FTP server program can be eliminated from the server host.

3.1.3 Transparent FTP Tunneling

SSH Tectia ConnectSecure provides transparent FTP tunneling which is the quickest way to secure file transfers. Both the original FTP client and server are retained and the file transfers are secured by encrypted tunnels.

Transparent tunneling of existing FTP connections

Transparent FTP tunneling provides a quick and easy way to secure FTP file transfers without the need to change the existing FTP scripts. Users can keep using the existing applications with their familiar IDs and authentication methods.

Full compatibility with FTP

Transparent FTP tunneling uses the Secure Shell v2 protocol to tunnel the existing FTP client and server connections providing full compatibility with existing unsecured FTP file transfer environment. Transparent FTP tunneling can be used to secure both interactive and unattended FTP sessions. Likewise, both active (initiated by FTP server) and passive (initiated by FTP client) FTP sessions are supported.

The existing FTP clients and servers are kept running, and they can continue performing their tasks, for example post-processing the transferred files.

Easy migration

Transparent FTP tunneling is an ideal solution for environments with thousands of complex FTP jobs with possible file transfer pre- and post-processing.

Transparent FTP tunneling also allows falling back to plaintext FTP, in case a Secure Shell tunnel cannot be established. This makes it possible to start migrating to secure file transfer usage immediately, and still be able to connect to the remaining FTP applications.

The principle of transparent FTP tunneling is shown in Figure 3.3.

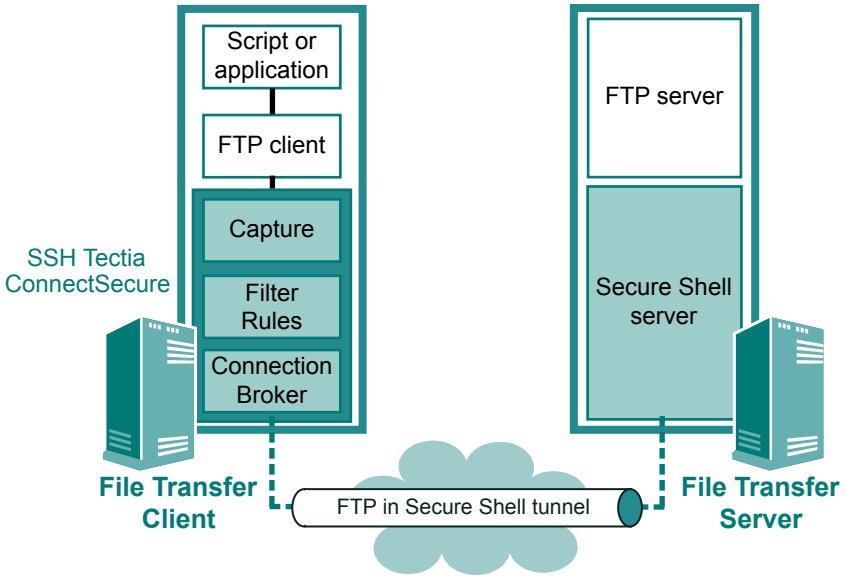

Figure 3.3. Transparent FTP tunneling

The following steps happen in transparent FTP tunneling:

1. An application, a script, or a user triggers a file transfer.

2. The FTP client in the File Transfer Client machine starts a file transfer to the FTP server in File Transfer Server.

3. The SSH Tectia connection capture module captures the connection before it leaves the client side. SSH Tectia ConnectSecure checks and applies the filter rules that specify which connections to capture. The filter rules are defined in the Connection Broker configuration. Connections can be captured based on the FTP application used and the destination address and/or port.

4. SSH Tectia ConnectSecure can extract the user name, password, and destination host name from the secured FTP application, and use them for authentication and connection setup with the Secure Shell server.

   The Connection Broker module creates an authenticated and encrypted Secure Shell tunnel to a Secure Shell server. The user can be authenticated with the FTP username and password, or with public keys. The Secure Shell server can be the FTP server specified in the original FTP request, or another server can be configured in the filter rules.

5. The secure tunnel is terminated at the Secure Shell server.

6. The Secure Shell server forwards the connection to the FTP Server, and the FTP server can continue with post-processing of the transferred files. If the FTP server is located on a third host, the connection from the Secure Shell server to the FTP server will be unsecured. This is why it is recommended that there is at least one Secure Shell server in each physically secured area, for instance in a machine room.

3.2 Enhanced File Transfer (EFT) with SSH Tectia ConnectSecure

With SSH Tectia ConnectSecure, the baseline SSH Tectia client/server solution functionality can be expanded to perform enhanced file transfer (EFT) operations. EFT offers higher encryption performance, more comprehensive manageability, APIs for application-level integration, and additional reliability features such as checkpoint/restart for file transfers.

SFTP APIs

Two complete client-side SFTP APIs (application programming interfaces), one for Java and the other for C, enable seamless integration of secure file transfer capability to applications and third-party file transfer management systems.

When the SFTP functions have been integrated to an application, file transfers can be triggered from the application, real-time information can be received as the file transfers proceed, and full access to return codes is provided. The destination server can be any Secure Shell server.

Separate API documentation is provided with the product. For a use case example, see Section 5.3.

Checkpoint/restart mechanism

The checkpoint/restart mechanism provides fault tolerance for large file transfers without performance penalties for increased user productivity, improved transfer reliability, and easier file transfer management. The destination server can be any Secure Shell server.

During file transfer, the checkpoint/restarting creates entries (files) to a checkpoint directory storing the state of the transfer at the specified time: file timestamp and size, the position in the file, etc. If the file transfer is then interrupted for some reason, the last known state of the transfer is saved in the checkpoint entry, and the file transfer can be restarted from the known position. After the file has been successfully transferred, the checkpoint entry is removed.

Streaming

When SSH Tectia ConnectSecure is used together with SSH Tectia Server, file transfer performance can be enhanced with streaming.

Streaming transfers file contents between SSH Tectia Server and the client-side using a separate data channel instead of the SFTP channel that carries the SFTP commands, thereby avoiding some bottlenecks of the protocol. All data transferred is still encapsulated into the Secure Shell transport connection, meaning that this enhancement does not have any security implications. The SFTP streaming protocol extension is fully backward-compatible with all earlier SSH Tectia Client and Server versions.

File prefixing

File prefixing prevents unintentional file usage at the destination before the file transfer is complete. It enables optimized third-party job scheduling and event-based triggering of file resending from destination directories.

The prefixing adds a prefix to a filename during the file transfer and thus renames it. The prefix is removed after the file has been successfully transferred and the file has its original name again.

File prefixing can be activated with any Secure Shell server as the destination server.

Extended MVS streaming in IBM z/OS systems

When SSH Tectia ConnectSecure is used together with IBM z/OS systems, optimized throughput can be provided with extended streaming that uses direct, one-step processing of MVS dataset transfers, and converts the file format and character set on-the-fly between the distributed hosts and IBM z/OS systems.

3.3 Secure Application Connections

The SSH Tectia client/server solution offers tunneling and secure shell tools as ways of securing data communications between any standard TCP-based applications. The `sshg3` command-line tools can be used interactively or in scripts.

Tunneling, or port forwarding, is a way of forwarding otherwise unsecured TCP traffic through SSH Tectia in encrypted format. You can secure for example POP3, SMTP, and HTTP connections that would otherwise be unsecured.

Tunneling provides encryption and strong two-factor authentication to third-party network client applications. SSH Tectia allows different forms of tunnels depending on the environment and type of usage of the workstations or user terminals.

Secure connectivity over Internet

Tunneling makes it possible to access e-mail from any type of Internet service irrespective of the access method (modem, GPRS, 3G, a DSL line, or a cable connection, or a hotel Internet service). As long as the users have a TCP/IP connection to the Internet, they can get their e-mail and access other resources from anywhere in the world securely.

This is often not the case with more traditional IPSec-based VPN technologies because of issues with traversing networks that implement Network Address Translation (NAT). This is especially the case in hotels. NAT breaks an IPSec connection unless special protocols such as NAT- Traversal are implemented on the client and gateway. A hardware gateway is usually also needed.

Non-transparent tunnels

SSH Tectia Client supports non-transparent application tunneling, which means that the tunneled applications need to be defined on the basis of the TCP ports they use. Applications with dynamic ports are not supported.

CryptiCore® support

The SSH G3 architecture and the high-speed CryptiCore® algorithms (Intel platforms) help in meeting the performance requirements of large-scale application access scenarios. CryptiCore enables up to 600 Mb/s application tunneling throughput in 1Gb networks with SSH Tectia Server.

3.3.1 Transparent TCP tunneling

SSH Tectia Client and SSH Tectia ConnectSecure can be used for transparent tunneling of TCP connections. Both SSH Tectia products can capture all network communication originating from client-server applications on the local workstation such as email clients, web browsers, and other software; and tunnel the connections to any Secure Shell server.

Transparent TCP tunneling feature can be used, for example, to enable company employees access their e-mail, the company intranet pages and shared files securely even when working outside the office.

SSH Tectia Client and ConnectSecure enable transparent TCP tunneling without the need to modify the tunneled applications. Hence, the main barrier for wider adoption of Secure Shell tunneling is eliminated, as there is no more need to reconfigure the local host addresses in the application client's network settings.

SSH Tectia Client and SSH Tectia ConnectSecure can secure network client applications that initiate connections to server applications using TCP communications. Other network protocols such as UDP are not supported. Note that applications that initiate connections from the server to the client are not supported.

Broad application support

Transparent TCP tunneling can be used to encrypt any TCP-based user client/server application including both commercial application software and internal legacy applications.

Authentication by applications

The client-server applications carry out their own authentication procedures, if any, the same way they would without the encrypted tunnel.

No application modifications needed

Transparent TCP tunneling requires no re-configuring of the application software or port settings. The Connection Broker configuration contains all necessary settings.

Easy configuration

The applications to be tunneled and the other tunneling settings are defined in the Connection Broker configuration with filter rules. The configuration can be made so that the SSH Tectia product uses the user name and the destination host definition from the tunneled application, so that it is not necessary to create a separate rule for each destination server. Both SSH Tectia Client and ConnectSecure can use the same configuration format.

Fine-grained policy control

Administrators can freely define application security policies including rules to tunnel, allow plaintext, or block specific client-side application connections. The flexible configuration interface provides administrators with multiple ways of specifying the tunneled applications; applications can be identified according to destination address and/or port, application name, or location of the application client binary.

The principle of transparent TCP tunneling is shown in Figure 3.4.

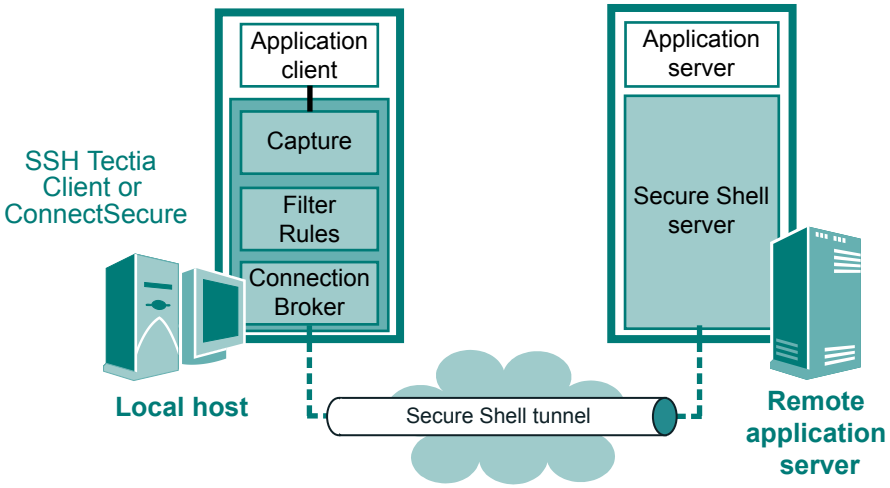

Figure 3.4. Transparent TCP tunneling

The following steps happen in transparent TCP tunneling:

1. A user or script on the local host starts an Application client that begins to establish a connection to the Application server.

2. The SSH Tectia connection capture module captures the connection before it leaves the client side. Only user processes are captured, and processes running with the `SYSTEM` account are passed through.

3. The SSH Tectia connection capture queries from the Connection Broker the filter rules that specify which connections to capture, and what filter rules to apply to the connection; whether to block it, let it pass directly, or to tunnel it securely through a Secure Shell server.

   The filter rules are defined in the Connection Broker configuration. Connections can be captured based on the application used and the destination address and/or port, or all application connections can be captured.

   If the connection requires tunneling, the Connection Broker creates a TCP listener as a local tunnel end point and the application connection is redirected to that local end point.

4. SSH Tectia Client or ConnectSecure can extract the user name, password, and destination host name from the secured application, and use them for authentication and connection setup with the Secure Shell server.

   The Connection Broker module creates an authenticated and encrypted Secure Shell tunnel to a Secure Shell server, which, if necessary, relays the traffic to the destination host. The Secure Shell server can be the application server specified in the original connection request, or another server configured in the filter rules.

5. The secure tunnel is terminated at the Secure Shell server.

6. The Application server is the end point of the connections. If the Application server is located on a third host, the relayed connection from the Secure Shell server to the application server will be unsecured. This is why it is recommended that there is at least one Secure Shell server in each physically secured area, for instance in a machine room.

3.4 Strong Authentication

The SSH Tectia client/server solution offers several methods for user and server authentication, and true strong authentication using either public keys or public-key certificates.

Server Authentication with Public Keys or Certificates

The SSH Tectia client-side components authenticate the Secure Shell server in order to verify that they are connecting to the correct server. Likewise, the Secure Shell server authenticates the client user. The server can be authenticated by either (plain) public-key authentication or certificate authentication. When certificate authentication is used, the public key is included in the certificate that the server sends to the client.

In (plain) public-key authentication, the server sends its public key to the client at the beginning of the first connection, and after the user has once verified and accepted the key, it is used in all future connections to that server.

In certificate authentication, the SSH Tectia client-side components rely on a trusted third party, a certification authority (CA) to verify the server's identity. The signature of the certification authority in the server certificate guarantees the authenticity of the server certificate.

User Authentication with Certificates and Public Keys

Client-side users can use certificates as proof of their identity. Certificates work like passports; the user prooves his or her identity to a certificate authority once using public keys, receives a certificate and from then on can authenticate using the certificate.

Public keys

Public-key authentication (without certificates) provides an easy-to-deploy and secure means of authenticating the users without the need to deploy and maintain a public-key infrastructure (PKI). Users will create key pairs for themselves, and upload the public keys to the server for verification.

Authentication agent

SSH Tectia Client and ConnectSecure incorporate authentication agent functionality that allows the caching of passphrases, eliminating the need to retype the passphrase each time when a connection is made. Passphrases are used in public-key authentication, which is more secure than password authentication. In addition, authentication can be "forwarded" to another machine, allowing administrators to hop from one server to another without the need to store private keys in multiple servers.

Passwords

SSH Tectia supports secure password-based authentication. Unlike in plaintext protocols such as Telnet and FTP, passwords are never sent in plaintext format over the network, thus eliminating the risk of password exposure to unauthorised parties.

X.509v3 certificates

SSH Tectia supports X.509v3 certificates for further security and scalability in large and dynamic network environments. Comprehensive support for IETF PKIX and PKCS standards ensures seamless interoperability with third-party PKI products.

Flexible certificate revocation

SSH Tectia supports both Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP) for centralized revocation of user credentials. CRLs are automatically fetched from a local file or by using HTTP or LDAP, depending on the local settings and the CRL Distribution Point extension in the certificate. CRLs can also be imported offline in legacy environments.

Certificate lifecycle management

SSH Tectia supports IETF PKIX standards (CMPv2) and Cisco Systems' Simple Certificate Enrollment Protocol (SCEP) for online certificate enrollment. Certificates can also be imported by using the PKCS#12 envelope format supported by most Certification Authorities (CAs). SSH Tectia has been integrated with Entrust PKI for transparent certificate lifecycle management in Entrust environments. Entrust support is available on Windows platforms.

Smart cards and PKI tokens

SSH Tectia Client and ConnectSecure support smart cards, USB tokens, and other PKI authentication devices by supporting PKCS#11 and MSCAPI for interfacing with authentication keys. Strong, two-factor authentication overcomes the inherent security issues of password authentication.

Host-based authentication on Unix

Host-based authentication is a form of delegated trust authentication, where the Secure Shell server trusts the Secure Shell client host to authenticate the user. The user is verified by a `suid` binary (ssh-signer) on the client host which then confirms the user identity to the server in a communication signed with a root-owned host key. The client host is authenticated strongly with public key cryptography, thus the authentication does not rely solely on a host IP address or domain name. The Secure Shell host-based authentication utilizes strong cryptography for host identity verification.

Keyboard-interactive

Keyboard-interactive is a standards-based method of integrating Secure Shell with third-party authentication mechanisms that are based on keyboard input, without the need to modify the client-side application (SSH Tectia Client). Keyboard-interactive is commonly used in conjunction with PAM and RADIUS on the server side.

PAM support

SSH Tectia Server supports Pluggable Authentication Module (PAM) for integrating with third-party authentication systems that have standards-based PAM libraries.

LDAP integration

SSH Tectia Server can utilize standards-based third-party LDAP directories as centralized user repositories. The keyboard-interactive method and third-party PAM modules for LDAP can be used for integrating SSH Tectia Server on Unix with LDAP directories.

RSA SecurID

SSH Tectia Client, ConnectSecure, and Server support RSA SecurID for strong, two-factor authentication. The keyboard-interactive method is used for providing the password from SSH Tectia Client or Connect-Secure to Server, which is integrated with the RSA Authentication Agent libraries for seamless interoperability.

RADIUS support

The RADIUS (Remote Authentication Dial-In User Service) protocol can be used with the SSH Tectia client/server solution for checking users' authentication and authorization information from a remote server. Keyboard-interactive is used for sending the password to SSH Tectia Server, which interfaces with the third-party RADIUS server such as Microsoft IAS or FreeRADIUS.

GSSAPI authentication (Kerberos)

Kerberos/GSSAPI authentication enables transparent, single sign-on alike authentication of SSH Tectia Client users. Once the user has logged on to the network and received the logon credentials, there is no need to type in the authentication credentials again through SSH Tectia Client user interface when accessing Secure Shell servers. Specifically, Kerberos/GSSAPI authentication enables the use of Windows domain authentication and Active Directory accounts with SSH Tectia (SSPI API in Windows).

OpenSSH key support

SSH Tectia Client, ConnectSecure, and Server support the legacy OpenSSH public-key format, eliminating the need for manual key conversions in multi-vendor Secure Shell environments. The key-compatibility feature also allows easy migration of OpenSSH environments to SSH Tectia.

Centrify DirectControl support

Integration of SSH Tectia with Centrify DirectControl enables secure host access while leveraging Active Directory-based identity management throughout multi-platform enterprise networks.

3.5 SSH Tectia Client/Server Solution Features

The following general features are available with all products of the SSH Tectia client/server solution.

Compliance with the IETF Secure Shell standards

The SSH Tectia client/server solution implements the Secure Shell (version 2) protocol as defined by the IETF Proposed Standard RFC specifications. SSH Communications Security is the original developer of Secure Shell and has been an active driver of the Secure Shell standardization in the IETF.

Comprehensive cryptographic support

The SSH Tectia client/server solution offers state-of-the-art encryption with broad support for symmetric ciphers including 3DES, AES, Arcfour, Blowfish, SEED, and Twofish. Supported message authentication and public-key algorithms include MD5, SHA-1, Diffie-Hellman, DSA, and RSA.

Hardware acceleration of encryption on Linux on IBM System z

SSH Tectia Server for Linux on IBM System z automatically uses hardware acceleration of encryption operations with the IBM-provided cryptographic hardware CPACF, if it is available. Hardware acceleration optimizes encryption performance and CPU usage, and it is used with the 3DES, SHA-1, and AES algorithms.

FIPS-certified cryptographic library

SSH Tectia Client, ConnectSecure, and Server incorporate a FIPS 140-2 certified cryptographic module to help ensure acceptance in government audits. The FIPS 140-2 Cryptographic Library has been validated for both Windows and major Unix platforms. The mode of the cryptographic library can be changed easily in the SSH Tectia Configuration GUI or by editing the configuration file.

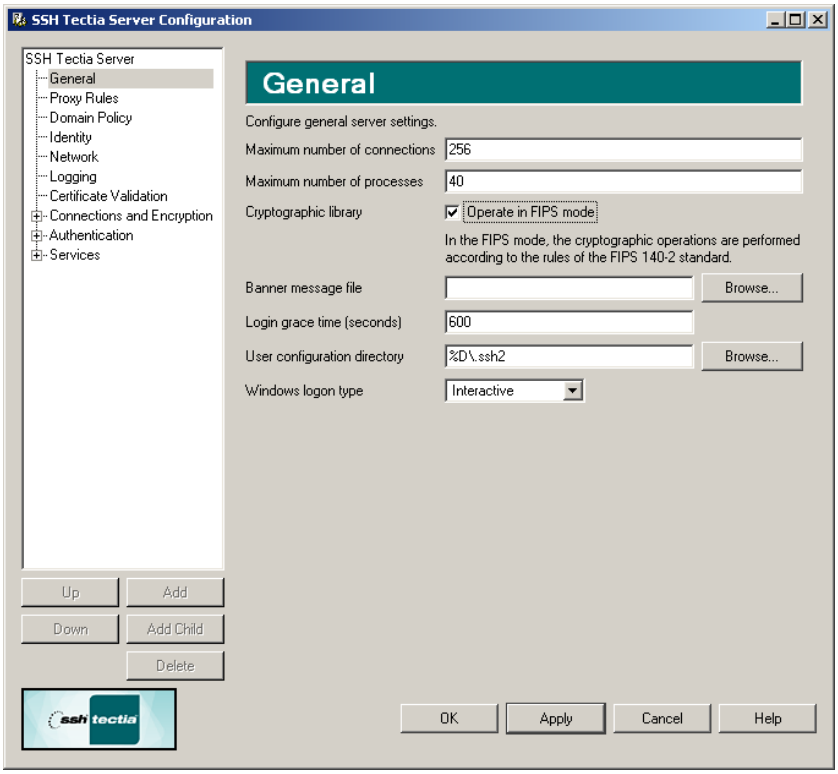

Figure 3.5. Activating FIPS mode on SSH Tectia Server

For a list of platforms on which the FIPS library has been validated or tested, see Section 6.5.

Versatile command-line tools

The SSH Tectia products include versatile command-line tools that can be used for remote login, remote command execution, and file transfer operations. These tools allow easy scripting of automated jobs such as secure file transfers or starting and stopping of services in remote locations.

Tunneling (port forwarding)

One of the key features of Secure Shell in addition to secure terminal access and secure file transfers is its ability to tunnel TCP-based application connections. The SSH Tectia products allow static application tunneling where application client connections are routed through the local TCP port, and then securely tunneled to a remote Secure Shell server.

Automatic tunneling

Before an application can be tunneled, a Secure Shell connection needs to be established. When using the automatic tunneling feature, the SSH Tectia client-side component listens to a specific port and establishes the encrypted connection automatically when the specific application is connecting to the local host port.

Firewall traversal

The SSH Tectia products support SOCKS (4 and 5) and HTTP proxy for accessing Secure Shell servers located behind firewalls.

Multi-channel support

Multi-channel support allows users to have multiple terminal sessions, file transfers, and application tunnels that are multiplexed to a single Secure Shell connection without the need to authenticate every session separately.

Configurable re-keying policies

Administrators can configure the renewal period for session encryption keys according to the security requirements.

3.6 High Performance with SSH G3

SSH G3 is a third-generation Secure Shell protocol implementation, which has been optimized for higher performance in demanding file transfer and application tunneling environments. The SSH G3 architecture provides unparalleled Secure Shell encryption throughput and scalability for large organizations.

Connection scalability

SSH G3 implements an `n x m` server process architecture for optimized server-side memory consumption and performance. While each server process (total amount n) can handle multiple (m) connections, the memory consumption per connection is considerably lower than in the second-generation Secure Shell implementations, making SSH Tectia an ideal solution especially for large-scale application tunneling.

Client-side Connection Broker

The Connection Broker is a key component in the SSH G3 architecture, handling all protocol and cryptographic operations. Client-side memory consumption is reduced since there needs to be only a single Connection Broker instance running per user. Security is also further improved by isolating all security-critical operations including authentication data handling in a single component.

Higher throughput

The SSH G3 architecture has been designed to minimize internal data handling such as data copy operations to minimize the throughput time in large file transfers.

Multi-threading

SSH G3 utilizes multi-threaded programming to fully leverage multi-processor servers for improved performance.

3.7 Ease of Use

SSH Tectia client/server solution is easy to install, and it can be configured using a GUI or by editing an XML configuration file.

Easy installation

The installation process of SSH Tectia products is effortless and requires no changes to the operating system. SSH Tectia Client can also be easily installed by the end users themselves if the security policy allows. Installations can also be managed with SSH Tectia Manager which is a separate product for centralized deployment and maintenance of all SSH Tectia software.

Drop-in replacement for Telnet and FTP

The SSH Tectia client/server solution provides easy and cost-effective means of securely replacing plaintext Telnet connections and file transfers in heterogeneous enterprise networks. The plaintext file transfer connections can be converted to secure SFTP or tunneled (encrypted) transparently to the users and their existing applications. In addition, SSH Tectia Client and ConnectSecure include a user interface similar to the widely used Telnet and FTP tools.

Drag-and-drop file transfers

An easy-to-use GUI for Windows allows users to securely drag-and-drop files between Windows and remote Unix/Linux/Windows/mainframe systems.

XML configuration file format

Demanding and complex security rules such as access control and authentication configurations can be easily set up by using the XML-based configuration file format. Administrators can use existing third-party XML parsers to efficiently view and edit the configurations settings.

Windows domain authentication

The SSH Tectia client/server solution can be integrated with Windows domain authentication by using Kerberos/GSSAPI for fully transparent user authentication. Once the users are logged on to the domain, there is no need for additional interaction for Secure Shell user authentication.

Configuration GUI

SSH Tectia Client, ConnectSecure, and Server offers an intuitive GUI for locally configuring all relevant Secure Shell settings needed for secure terminal access, file transfers, and static application tunneling. A separate product, SSH Tectia Manager available for centralized Secure Shell configuration management and security policy enforcement.

3.8 Centralized Management with SSH Tectia Manager

SSH Tectia Manager is a separate product that offers centralized management and powerful reporting capabilities for the SSH Tectia client/server solution. Hosts running SSH Tectia Client, ConnectSecure or Server can be placed under control of SSH Tectia Manager, and then the SSH Tectia software can be installed, upgraded, configured and monitored in a centralized manner from the Management Server without any end-user intervention. Security reports can be produced for example for compliance audits.

SSH Tectia Manager enables effective security management of large and heterogenous environments as it provides a centralized repository and distribution point of SSH Tectia software and policy configuration. This way the amount of time and manual work required by the maintenance operations can be minimized, which reduces the total cost of ownership of the environment.

SSH Tectia Manager consists of Management Server, where the configurations and host data are stored, and Management Agents that are installed on the remote hosts and report all relevant SSH Tectia data from the hosts to the Management Server.

Centralized SSH Tectia Deployment and Upgrades

Centralized mass deployment of SSH Tectia software components and upgrades dramatically reduces the time and resources required by environment maintenance tasks and minimizes costly system downtime. Fast multi-platform deployment ensures consistency of security software running throughout the environment.

Centralized SSH Tectia Configuration Management

An intuitive administration GUI simplifies the intricacies of security configuration settings and configuration file syntax. Fast multi-platform distribution throughout the environment enables fast deployment of policy changes and minimizes service downtime. Enforced security setting consistency ensures a high level of security throughout the organization.

Centralized Server Host Public Key Distribution and Server Host Certificate Management

SSH Tectia Manager provides automated centralized distribution of SSH Tectia Server host keys, removing unnecessary "host key changed" alerts, and enabling the user to distinguish real security threats. Server

host keys are now kept up-to-date throughout the entire environment and transparent authentication is ensured also for the first connection to a server.

For large and dynamic environments, the built-in certification authority functionality provides further scalability for host authentication management. Centralized deployment of PKI settings further reduces management costs also when external PKI is in use.

Centralized SSH Tectia Server Log Gathering and Viewing

Centralized SSH Tectia Server log gathering provides a comprehensive view into the SSH Tectia environment. Flexible filtering enables tracking of details such as unsuccessful login attempts, enabling fast reaction to anomalies and error situations.

Environment Status Monitoring, Auditing And Reporting

SSH Tectia Manager provides tools for monitoring the status of the environment, for example the deployed SSH Tectia versions. In addition, administrators can run reports to prove to auditors that the security software is operational according to the corporate security policy.

SSH Tectia Manager is described in *SSH Tectia Manager Product Description*.

3.9 Compatibility with IBM Mainframes

SSH Tectia product family includes a separate solution for IBM mainframes. SSH Tectia Server for IBM z/OS is installed on mainframes, where it provides the same secure shell services as the SSH Tectia Server on other platforms, including secure file transfers, secure application connectivity and secure remote access.

SSH Tectia Client and ConnectSecure are capable of connecting to the SSH Tectia Server for IBM z/OS, and data can be transferred to and from the mainframes. SSH Tectia Client and ConnectSecure provide the following mainframe-specific features.

MVS dataset handling

When used in conjunction with SSH Tectia Server for IBM z/OS, users of SSH Tectia Client and ConnectSecure can list IBM MVS (Multiple Virtual Storage) datasets as files and folders, facilitating seamless cross-platform file transfer between mainframe and non-mainframe systems.

Extended MVS streaming

When used in conjunction with SSH Tectia Server for IBM z/OS, users of SSH Tectia ConnectSecure can use extended streaming for direct, one-step processing of MVS dataset transfers. Extended streaming provides optimized throughput, and on-the-fly conversions for file format and character sets between distributed hosts and IBM z/OS systems.

Transparent FTP tunneling

SSH Tectia Server and ConnectSecure can capture and encrypt the file transfer connections to and from IBM Mainframes. Transparent FTP tunneling provides a quick and easy way to secure FTP file transfers without the need to change the existing FTP scripts or applications.

FTP-SFTP conversion

SSH Tectia ConnectSecure provides FTP-SFTP conversion, which allows easy and cost-effective replacement of plaintext file transfers in large enterprise environments. Existing FTP scripts and client applications need no modifications. Only the FTP server will be eliminated and an SFTP server is used instead; the SSH Tectia Server for IBM z/OS provides the SFTP server function.

ASCII/EBCDIC code set translation

Full and configurable ASCII/EBCDIC conversion is supported as well as configurable CONVXLAT conversion tables for seamless cross-platform compatibility between IBM z/OS and Unix/Linux/Windows hosts.

SSH Tectia Server for IBM z/OS is described in *SSH Tectia Server for IBM z/OS Product Description*.

Chapter 4 Architecture

This section introduces the components included in the SSH Tectia client/server solution architecture, the configuration tools, and how the server and user authentication work.

The SSH Tectia client/server solution utilizes client-server architecture. By default, the server listens to TCP port 22, which has been officially assigned for Secure Shell, and clients initiate connections to this port. The listener port can be changed in the server configuration.

The SSH Tectia client/server solution works with any type of Internet (TCP/IP) connection - ADSL, ISDN, modem, Ethernet, WLAN, PPPoE - thus making it widely applicable, totally independent of the network topology, and independent of network address translations or other features that may burden some other security solutions.

4.1 SSH Tectia Solution Components

The SSH Tectia client/server solution consists of five products:

- SSH Tectia Client - provides the basic Secure Shell client features and tools.
- SSH Tectia ConnectSecure - provides robust enterprise-level secure file transfer features in addition to the basic Secure Shell client features; and it is meant for server host installations.
- SSH Tectia Server - provides the Secure Shell server features and tools.
- SSH Tectia Server for Linux on IBM System z - provides both the server-side and the client-side Secure Shell components in installations on Linux on IBM System z platforms.
- SSH Tectia Server for IBM z/OS - provides both the server-side and the client-side Secure Shell components in installations on IBM mainframes. SSH Tectia Server for IBM z/OS is introduced in its own *Product Description*.

The SSH Tectia products running on Unix and Windows can be centrally managed with SSH Tectia Manager. However, SSH Tectia Manager cannot be used to manage the mainframe-installed servers, SSH Tectia Server for Linux on IBM System z and SSH Tectia Server for IBM z/OS.

The components are described in the following chapters and illustrated in Figure 4.1.

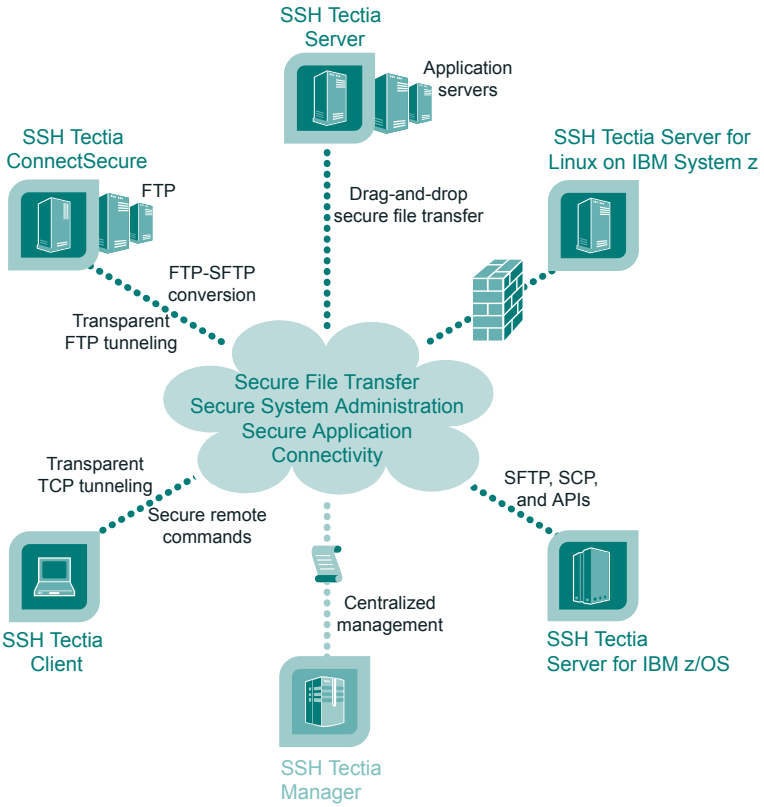

Figure 4.1. The components of the SSH Tectia client/server solution

4.1.1 SSH Tectia Client

SSH Tectia Client provides secure interactive file transfer and terminal client functionality for remote users and system administrators for accessing remote hosts running SSH Tectia Server or other Secure Shell server.

SSH Tectia Client provides easy-to-use graphical user interfaces for file transfers and for configuring the Connection Broker settings on Windows. The intuitive file transfer window includes separate views for the file folders in the local and remote computers, as well as transfer progress and history views.

SSH Tectia Client also includes advanced command-line tools for system administrators to set up secure automated file transfers, and support for outgoing and incoming application tunneling, such as X11 forwarding. On Windows, SSH Tectia Client supports also transparent TCP tunneling.

4.1.1.1 Connection Broker

The Connection Broker is an integrated component of SSH Tectia Client and SSH Tectia ConnectSecure. The Connection Broker handles all cryptographic operations and authentication-related tasks on the client side.

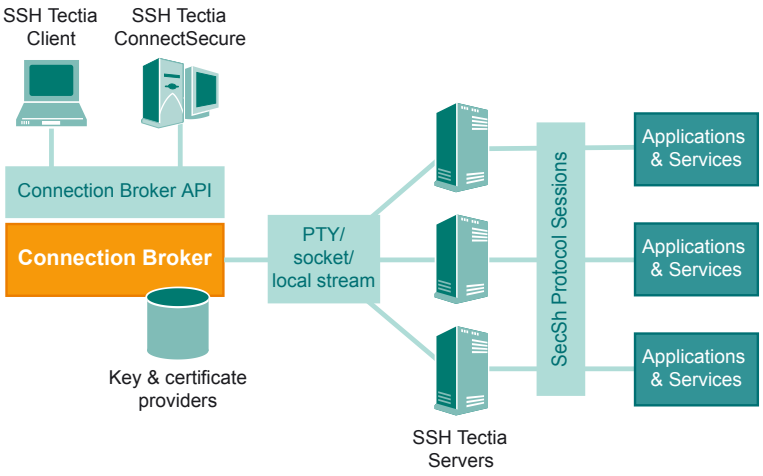

Figure 4.2. Connection Broker architecture

The connections of the Connection Broker to other applications is shown in Figure 4.3.

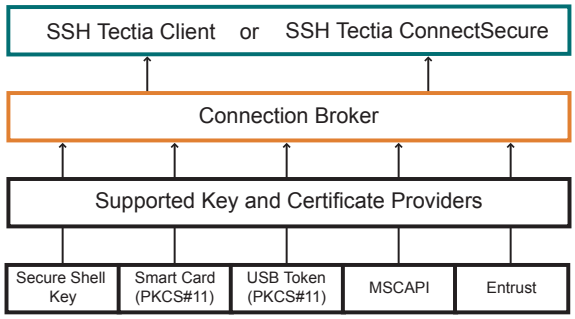

Figure 4.3. Connection Broker connections

The Connection Broker supports the following key and certificate providers:

- MSCAPI: Microsoft Crypto API, a standard cryptographic interface in Microsoft Windows-based systems.
- PKCS#11: Connection Broker supports cryptographic tokens based on PKCS#11 (v2.x).

  The PKCS#11 Public-Key Cryptography Standard specifies an API called Cryptoki to devices that hold cryptographic information and perform cryptographic functions. For more information, see the RSA Laboratories web page at http://www.rsasecurity.com/rsalabs/pkcs/pkcs-11/index.html.

- Entrust: By using the Entrust provider, SSH Tectia can utilize keys and certificates stored in an Entrust profile file (`.epf`). The initialization file includes the basic Entrust PKI configuration (for example the certification authority (CA) address).

4.1.2 SSH Tectia ConnectSecure

SSH Tectia ConnectSecure secures server-to-server connections, but it also acts as a Secure Shell client, providing enterprise-class file transfer features in addition to the basic client features. SSH Tectia ConnectSecure has been designed to enable secure and reliable high-performance file transfers and tunneling services in heterogeneous enterprise environments. SSH Tectia ConnectSecure provides also command-line tools for securing non-interactive and automated file transfers, and the client-side of APIs for integrating the SSH Tectia products to applications.

SSH Tectia ConnectSecure can connect to any standard Secure Shell server. When SSH Tectia Server is used as the counterpart, also the enhanced file transfer features are available.

SSH Tectia ConnectSecure provides SFTP tools, FTP-SFTP conversion, and transparent FTP tunneling. Transparency means that the end-user experience will not change and the existing file transfer applications require no modifications, while all configuration is done on the Connection Broker. For information on the Connection Broker, see Section 4.1.1.1.

In addition, most TCP-based in-house and commercial applications can be secured by tunneling them transparently on specified Windows platforms. On SSH Tectia ConnectSecure, transparent TCP tunneling is always available, and on SSH Tectia Client it is an optional feature on Windows. See the supported operating systems in Section 6.3.

SSH Tectia ConnectSecure encrypts the TCP/IP connections between two computers. The encryption of the connection is done on the application layer, which means that the security provided by SSH Tectia ConnectSecure is available regardless of the network connection speed or type. It is an ideal solution for providing transparent protection for business application traffic between corporate users' workstations and servers.

SSH Tectia ConnectSecure can be placed under centralized management provided by SSH Tectia Manager.

4.1.3 SSH Tectia Server

SSH Tectia Server is the server-side component for deploying secure file transfers, secure application connectivity and secure remote administration services over unsecured networks.

SSH Tectia Server is a robust, flexible and field-tested server implementation of the Secure Shell protocol. Its technology has been the choice of numerous large corporations, banks, financial organizations, and governments around the world.

SSH Tectia Server provides strong user authentication, traffic encryption/decryption, and traffic integrity checking, and out-of-the-box interfaces to integrate leading third-party authentication or authorization systems (such as RSA SecurID). There is also an easy-to-use graphical user interface for configuring SSH Tectia Server.

The SSH Tectia Server product is available in three versions designed for different platforms:

SSH Tectia Server

SSH Tectia Server is available for Unix and Windows computing platforms, such as Sun Solaris, IBM AIX, HP-UX, SUSE and Red Hat Linux, and Microsoft Windows. On Unix platforms, the server license includes also SSH Tectia Client.

SSH Tectia Server offers all Secure Shell server functionalities, including secure terminal, SFTP, and tunneling. When SSH Tectia ConnectSecure is acting as the counterpart, SSH Tectia Server supports also the enhanced file transfer (EFT) features, such as streaming for fast transfer of large files. SSH Tectia Server on Unix and Windows can be managed centrally by SSH Tectia Manager.

SSH Tectia Server for Linux on IBM System z

SSH Tectia Server for Linux on IBM System z has been designed for Linux running on IBM System z platforms. It provides the same services as SSH Tectia Server on Unix plus it supports hardware acceleration of encryption operations. SSH Tectia Server for Linux on IBM System z contains also the client-side components that provide the basic Secure Shell client features and tools. SSH Tectia Server for Linux on IBM System z cannot be managed centrally by SSH Tectia Manager.

SSH Tectia Server for IBM z/OS

SSH Tectia Server for IBM z/OS has been designed for z/OS platforms running on IBM mainframes. It provides the same services as SSH Tectia Server on Unix, and contains also client tools that support FTP-SFTP conversion, transparent FTP tunneling, and enhanced file transfer (EFT) features. SSH Tectia Server for IBM z/OS cannot be managed centrally by SSH Tectia Manager.

4.2 SSH Tectia Solution Configuration

The SSH Tectia client/server solution allows the administrator to modify the configuration settings, and to add various rules that control the operations of the SSH Tectia products and the secured connections. Both Windows and Unix versions have the same configuration file format in local configurations.

The administrator can view and edit the configurations either in XML format (with a text editor such as Emacs, vi, or an XML editor) or through a graphical user interface (GUI). The Configuration GUIs are similar on Windows and Unix platforms, but on Unix, the final looks of the GUI depend on the Unix version used. On Windows, SSH Tectia Client and ConnectSecure have an additional File Transfer GUI for handling secure file transfers.

The SSH Tectia client/server solution can also be placed under control of SSH Tectia Manager. Then the SSH Tectia products are managed centrally, and configuration sets can be created and stored on Management Server. Hosts running SSH Tectia Client, ConnectSecure, and Server can be arranged in host groups, and the configurations can be deployed jointly to all hosts in a group.

When the SSH Tectia products are managed centrally through SSH Tectia Manager, SSH Tectia Server and Connection Broker configuration GUIs restrict local configurations, which eliminates the possibility of unintentional user modifications through GUI.

There are different graphical user interfaces (GUI) available for the SSH Tectia client/server solution products:

- **Terminal GUI** for Secure Shell operations on SSH Tectia Client and ConnectSecure
- **SSH Tectia Configuration GUI** for handling the Connection Broker configuration on SSH Tectia Client and ConnectSecure (and SSH Tectia MFT Events) on all platforms
- **SSH Tectia Server Configuration GUI** for the server on all platforms

SSH Tectia Client and ConnectSecure use the same terminal window which offers several ways to open secure connections and to access the SSH Tectia services. See Figure 4.4.

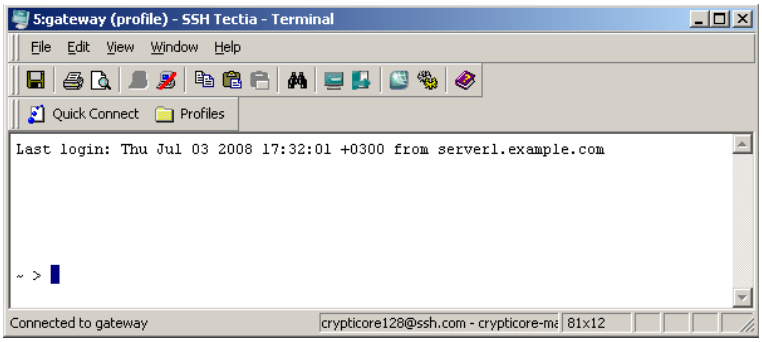

Figure 4.4. The SSH Tectia terminal window

SSH Tectia Client, ConnectSecure, and SSH Tectia MFT Events use the Connection Broker configurations which can be made through the SSH Tectia Configuration GUI; see Figure 4.5.

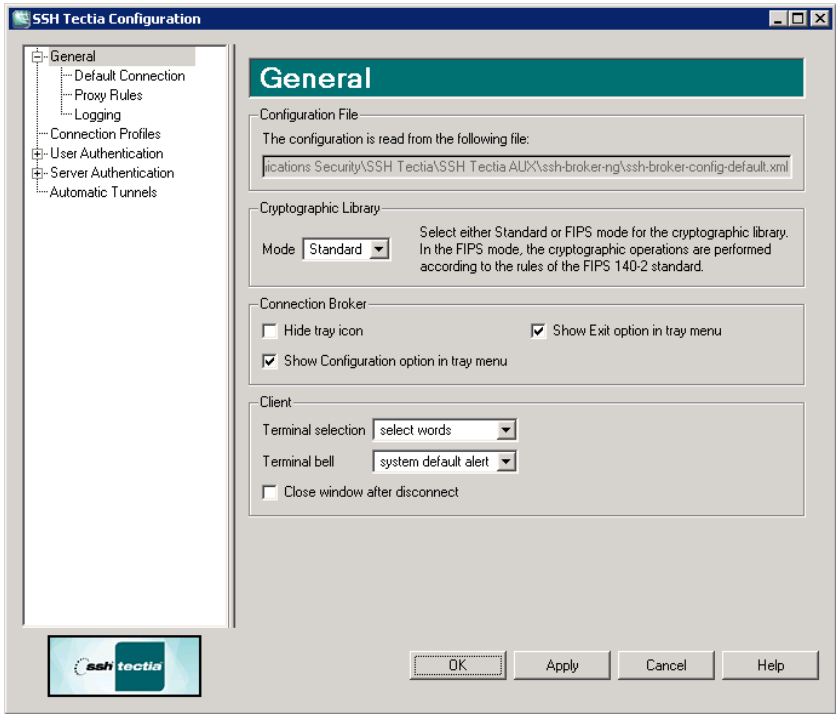

Figure 4.5. SSH Tectia Configuration GUI - General settings view

SSH Tectia Server can be configured through the SSH Tectia Server Configuration GUI; see Figure 4.6.

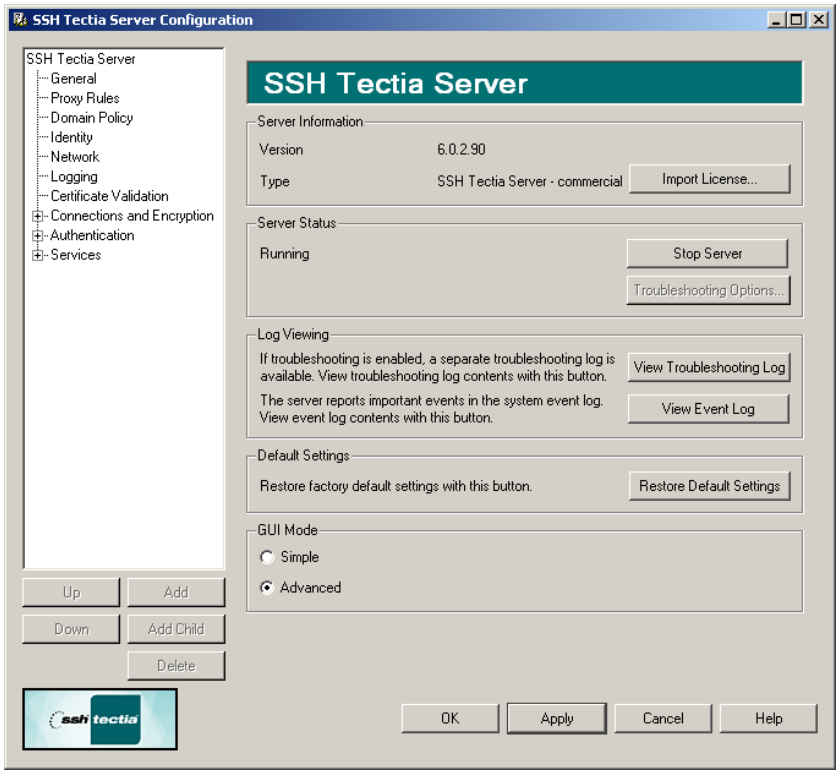

Figure 4.6. SSH Tectia Server Configuration GUI main page

4.3 Authentication

The SSH Tectia client/server solution provides mutual authentication for the server and the client user. The SSH Tectia client components authenticate the server and the SSH Tectia Server authenticates the user's identity.

All data for identification and authentication purposes is exchanged in encrypted format between the client and server.

User authentication has always been a strong asset of Secure Shell. Multiple methods have been standardized for user authentication and new methods can be implemented easily to the protocol. Secure Shell provides confidentiality for user authentication. User identity is not revealed to eavesdropping parties since the user is authenticated only after the connection has been secured.

For reference information on the user authentication methods, see technical note *SSH Tectia Client/Server User Authentication Methods* at http://www.ssh.com/resources/.

4.3.1 Server Authentication

SSH Tectia uses cryptographic authentication for Secure Shell server hosts. Each server has a cryptographic key pair (a public key and a private key) that identifies the server. Whenever a Secure Shell client connects to a Secure Shell server, the server authenticates itself to the client cryptographically. This ensures that encryption and integrity protection are provided end-to-end between the client and the intended server. Server authentication also eliminates the danger of certain cryptographic attacks, especially man-in-the-middle attacks.

For the cryptographic authentication to work, the client must know the server's public key so that it can securely authenticate the server. The public key of the server must be distributed to each client. The private key of the server is never sent anywhere outside the server computer, but it is used by the server to create a digital signature that can then be verified by the client using the public key.

SSH Tectia Manager can be used to automate the distribution and maintenance of the server public keys (host keys).

Secure Shell authenticates the Secure Shell server service of the server host. A host may run several Secure Shell server listeners on different ports. On Unix, each server can have a unique identity, if desired. However, typically there is only one Secure Shell server listener. If separate policies are needed or different services need to be offered for multiple use cases on a particular host, they can be defined dynamically in the SSH Tectia Server configuration.

The server is authenticated with a digital signature based on a DSA or RSA public-key algorithm. Each server must have a public-private key pair. In implementations without support for certificates, clients refer to a local database of trusted server public keys.

Secure Shell also supports certificate authentication. Servers can authenticate themselves to the client with X.509 v3 certificates. When certificates are used, the client does not need to have a local database of trusted server public keys or server certificates. Instead, just the few trusted CA (certification authority) certificates are stored on the client, and the client trusts the servers whose certificates are signed by a trusted CA and certificate contents match the server hostname. Certificates provide scalability for authentication.

The Secure Shell server may have multiple identities (one DSA key, one RSA key, one DSA certificate, and one RSA certificate). During the key exchange in the Secure Shell connection, the Secure Shell client and server agree on which identity will be used in server authentication. SSH Tectia Client and ConnectSecure prefer certificates over keys if trusted CA certificates have been configured, and otherwise DSA keys over RSA keys.

4.3.2 User Authentication

Different methods can be used to authenticate users in SSH Tectia. These authentication methods can be used separately or combined, depending on the level of functionality and security you want.

By default, the SSH Tectia client/server solution uses these user authentication methods:

- GSSAPI
- public-key
- password authentication
- keyboard-interactive

Public-key and certificate authentication are combined into the public-key authentication method.

The most commonly used user-authentication methods are password, public-key, and host-based authentication. In public-key authentication, the users upload their public key files to the server and edit a configuration file. The server thus has a database of user public keys, similar to the client having its database of server public keys.

When certificates are used in user authentication, the user does not need to upload any public key files to the server prior to the connection, and the server does not need to have a database of user public keys or certificates. The server validates the user certificate by using the CA certificates that the server has been configured to trust and authorizes the login based on the user certificate contents.

Keyboard-interactive is not an authentication method in itself, but more like a common interface to various other authentication methods that are based on keyboard input. Password authentication, RSA SecurID, PAM (Pluggable Authentication Module), and RADIUS are examples of authentication methods that can be used over keyboard-interactive.

The highest security is achieved by using token-based certificate authentication where the certificate and the private key are stored on a cryptographic token, such as a smart card. Secure Shell supports also several other strong authentication methods, including the proprietary RSA SecurID.

Chapter 5 Use Cases

This chapter introduces typical use cases of the SSH Tectia client/server solution. The use cases show examples of security services that you can quickly and effectively implement in typical enterprise environment.

About replacing unsecured FTP with secure file transfer, see Section 5.1, Section 5.2, and Section 5.3.

About securing application connections, see Section 5.6 and Section 5.7.

About securing remote administrator connections, see Section 5.8 and Section 5.9.

5.1 Replacing Plaintext FTP with FTP-SFTP Conversion

SSH Tectia ConnectSecure offers an easy way to secure plaintext FTP connections with a feature called FTP-SFTP conversion.

When FTP-SFTP conversion is enabled on SSH Tectia ConnectSecure, it automatically captures all FTP connections initiated on the client side and converts the data to use the Secure File Transfer Protocol (SFTP), instead. The transferred files are sent to a Secure Shell SFTP server in encrypted format.

SSH Tectia ConnectSecure should be installed on the same host with the FTP client, and a Secure Shell server must be installed on the same host with the original FTP server.

FTP-SFTP conversion can be configured to pick the user name, password, and destination host directly from the secured FTP client, and use them to open the secured communication channel. This removes the need for any additional configuration modifications or changes to the original FTP scripts or applications. In the Connection Broker configuration, this is done simply with one rule that can fit all FTP connections.

With SSH Tectia ConnectSecure on Windows, the FTP-SFTP conversion is enabled by default. On Unix platforms, it can be requested per FTP session.

When the FTP-SFTP conversion is used, there is no need for a plaintext FTP server, as the connection is made to an SFTP server instead. This requires that any post-processing done by the FTP server must be redirected to be performed elsewhere.

SSH Tectia ConnectSecure makes it easy to get started with the FTP replacement even in an environment where all FTP servers cannot be removed immediately. For example, there may be need to connect to a third-party FTP server every now and then, even though company-internal file transfers are handled in secure SFTP mode. SSH Tectia ConnectSecure has an option to allow fallback to plaintext FTP in case the secure SFTP connection cannot be established. This way the SFTP format is used always when possible, but connections to the remaining FTP servers are still available.

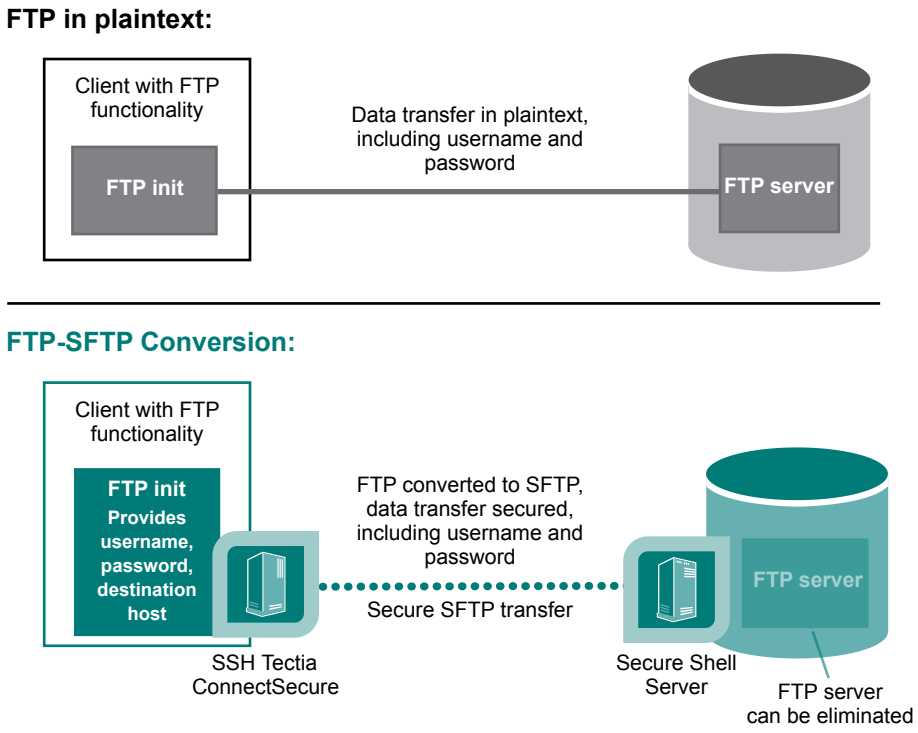

Figure 5.1. Using FTP-SFTP conversion

5.2 Securing Plaintext FTP with Transparent FTP Tunneling

Plaintext FTP is an inherently unsecured, but a widely used method of transferring files. SSH Tectia ConnectSecure offers an easy way to secure file transfer connections with transparent FTP tunneling. This feature is most useful when there is need to secure lots of FTP scripts.

Transparent FTP tunneling allows the FTP service to use the existing scripts and applications as they are, so to the users and applications the SSH Tectia FTP tunneling happens transparently. As the existing FTP applications are left running, for example the FTP servers can keep performing all their designated post-processing jobs as earlier.

Transparent FTP tunneling captures the connections that use the FTP protocol and tunnels them in encrypted format via a Secure Shell server to the FTP server. Transparent FTP tunneling can be configured to pick the user name, password and destination host directly from the FTP client, and use them to open the secured

communication channel. In the Connection Broker configuration, this is done simply with one rule that can fit all FTP connections.

The users can define connection profiles to perform transparent FTP tunneling of certain connections, or they can request the tunneling per FTP connection on command line.

For end-to-end security, SSH Tectia ConnectSecure should be installed on the same host with the FTP client, and a Secure Shell server should be installed on the same host with the FTP server. If end-to-end security is not required, the FTP server can also reside on a third host.

The FTP server side can be on any platform, Unix, Windows or mainframe. SSH Tectia Server for IBM z/OS works ideally with SSH Tectia products, but supports any SSH2-capable Secure Shell servers.

Transparent FTP tunneling can be used to secure both interactive and unattended FTP sessions. It also provides an option to fall back to plaintext FTP for easier migration.

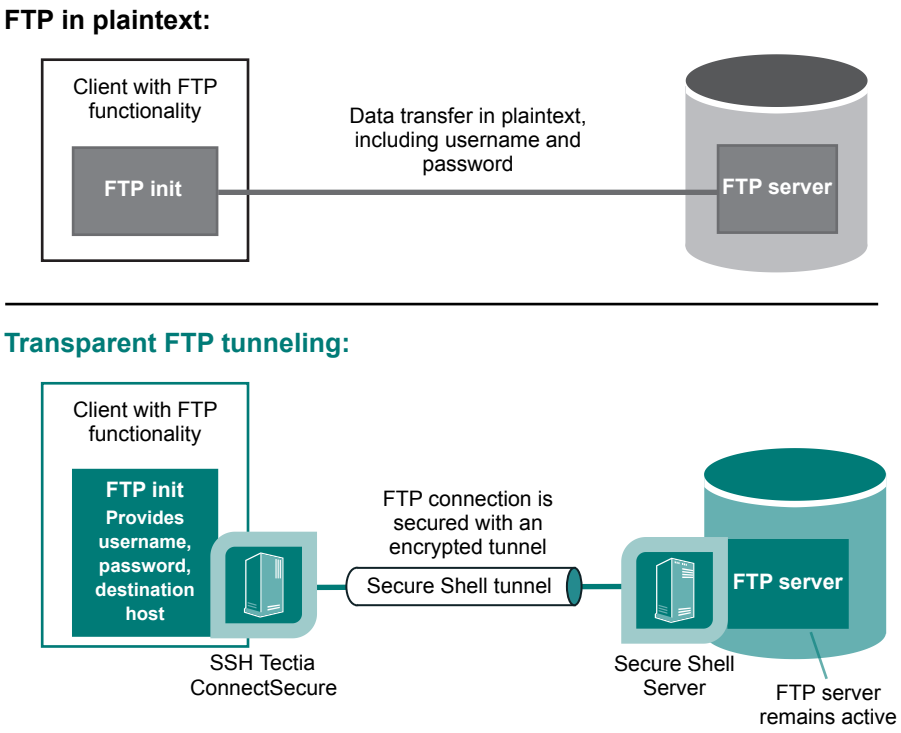

Figure 5.2. Using transparent FTP tunneling

5.3 Integrating an Extranet Application with SFTP

SSH Tectia ConnectSecure can be integrated to an existing application through the SFTP API, for example to an application responsible for data warehouse feeds, extranet applications, Windows desktop applications, or third-party file transfer management applications. The counterpart can be SSH Tectia Server or another Secure Shell server.

In this scenario, an end user needs to be able to download files from servers that are located on the Internet. However, direct access from the user workstations to those servers is not possible because of security reasons. Thus, the end users use a web access control application located on the extranet (DMZ). This application manages access rights to individual servers and files.

When the end user has selected the files to download, the web application triggers file transfer through the API. SSH Tectia ConnectSecure then transfers the files to the extranet and the user is provided HTML links to the files.

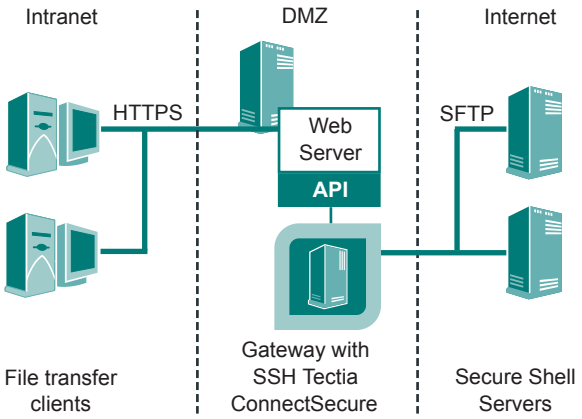

Figure 5.3. Integration through SFTP API

With this solution, end users are not accessing external services directly and it is easy to manage the access rights from a centralized web server. External parties do not have to modify their systems but can rely on the trusted Secure Shell standard.

5.4 Securing Gateways with SSH Tectia ConnectSecure

SSH Tectia ConnectSecure can be installed on a gateway host that mediates the connections for example between intranet applications and remote servers. There is no limit to the number of remote servers, and they can be distributed geographically.

SSH Tectia ConnectSecure can connect to any standard Secure Shell server, for example OpenSSH or VanDyke servers.

When SSH Tectia ConnectSecure is installed on the gateway host, no SSH Tectia software is necessarily required on the application hosts, if the intranet between them and the gateway is otherwise secured.

SSH Tectia ConnectSecure can be configured to secure all existing applications the administrators use to connect to the remote hosts, such as Telnet, Virtual Network Connection (VNC), FTP, Remote Desktop (RDP) and others. With the transparent TCP tunneling feature this can be made simply with a single filter rule set in the Connection Broker configuration. In the rule we just define that SSH Tectia ConnectSecure uses the

user name and the destination host name directly from the connection-originating application. This saves the effort of defining a separate connection profile for each destination host.

Once the transparent TCP tunneling setup is active, the administrators can keep using the existing tools without any modifications, and all connections from the gateway to the remote servers will be securely tunneled.

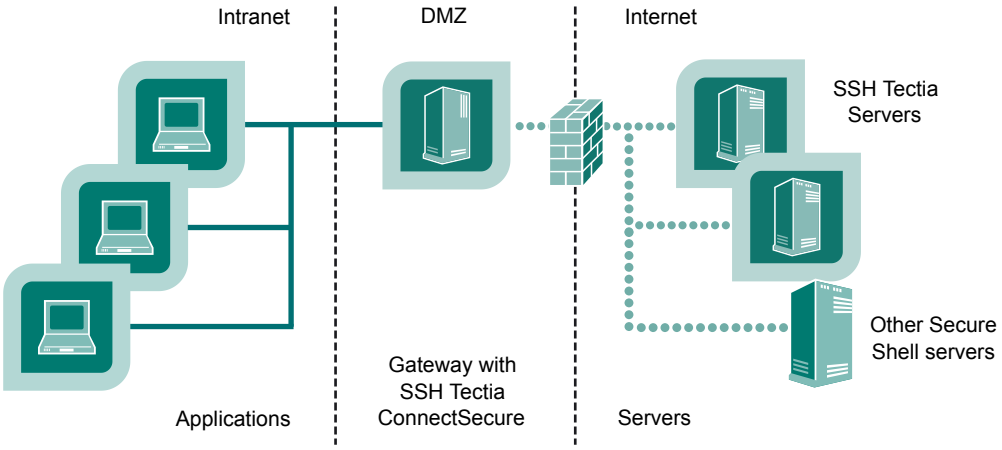

Figure 5.4. Securing gateway traffic with SSH Tectia ConnectSecure

5.5 Hardening Application Connectivity with Transparent TCP Tunneling

When SSH Tectia is used only to secure business applications with transparent TCP tunneling, it is not always necessary to implement strong user authentication with the SSH Tectia client/server solution. If it is acceptable from the security policy point of view to rely on the security of the application's own login mechanism, there is no need to require end users to perform double login (first to SSH Tectia Server, then to the application itself).

In this use scenario, the added value created by SSH Tectia is:

- Confidentiality and integrity is provided to application traffic.
- Passwords used for application login are encrypted in transit.

Note that in this use scenario SSH Tectia may be used in conjunction with a single sign-on (SSO) solution, which eliminates the need to sign on separately to each application.

User-specific authentication can be avoided by creating a common global account for a group of users, with rights to establish tunnels only (specifically no terminal or file access is allowed). The corresponding username and password can then be distributed with SSH Tectia Manager to those SSH Tectia Client and ConnectSecure users that need to access business applications running on the SSH Tectia Servers. SSH Tectia Client and ConnectSecure can then automatically connect to the server with the common user group credentials without the need to prompt the user for any login credentials. Therefore, from the end-user point of view there is no

visible additional authentication. But there is no true secure additional authentication either, as the global user account and password is shared between the users. Strong and transparent user authentication can be implemented with public key authentication, for example.

Figure 5.5 shows a network diagram of this use scenario.

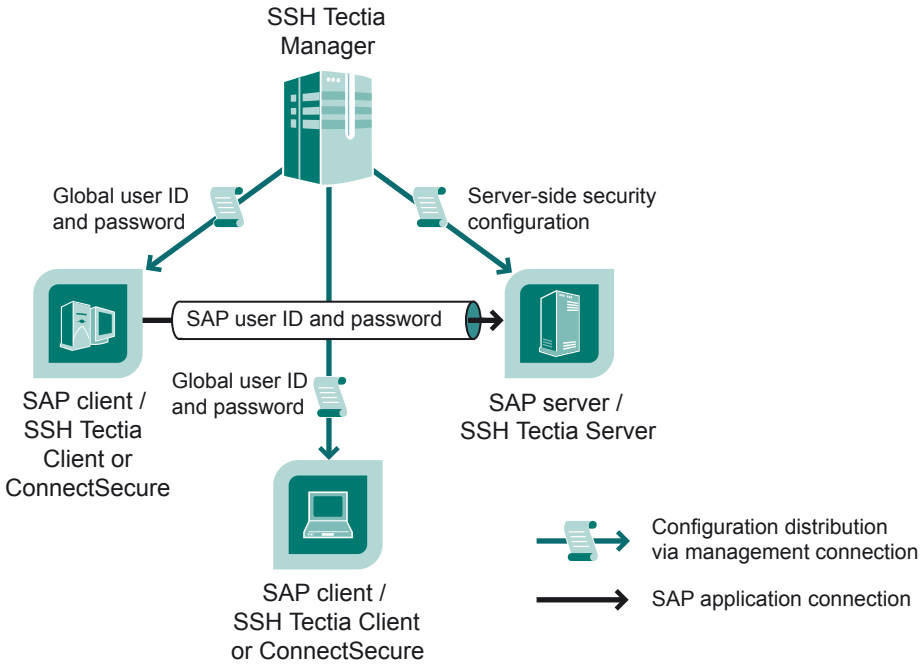

Figure 5.5. Using transparent TCP tunneling

5.6 Secure Application Login with Kerberos/GSSAPI

When both the client and server are located in the same Windows (NT or Active Directory) domain, it is possible to integrate Windows Domain logon to the SSH Tectia client/server solution. This means that when a user logs on to a Windows Domain, the user gets a "ticket" that can be used for authenticating the user to SSH Tectia Server. The authentication procedure is then non-interactive; the user is not prompted to enter a password when SSH Tectia Client or ConnectSecure are connecting to SSH Tectia Server.

GSSAPI authentication can also be used in a mixed Windows/Unix environment. On Unix, GSSAPI interoperates with standard Kerberos implementations that provide a GSSAPI mechanism.

Active Directory/Kerberos is used in the Windows 2003 Domains.

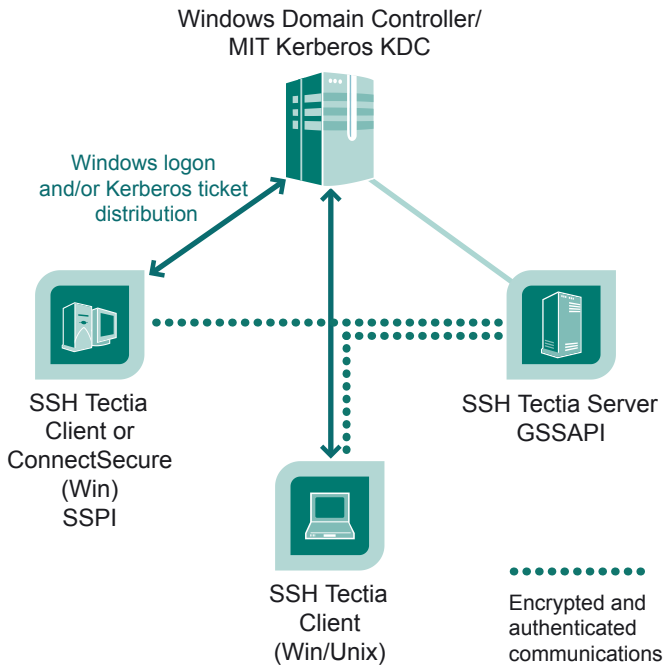

Figure 5.6. Secure application connectivity through GSSAPI

5.7 Remote Access through Nested Tunnels

SSH Tectia Client and ConnectSecure can be used to access Secure Shell servers on the corporate intranet from a remote location through nested tunnels. The transparent TCP tunneling feature is used to encrypt the application connections.

The first tunnel ends at the edge of the corporate intranet, and nested tunnels within the first tunnel continue to SSH Tectia Servers located inside the intranet. Only one port needs to be open in the firewall. It is also possible to chain the nested tunnels within the intranet if necessary.

This helps to achieve end-to-end security without using NAT and a private IP inside the corporate network. See Figure 5.7.

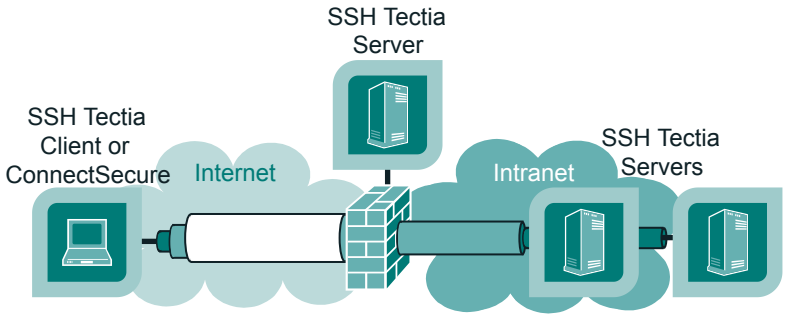

Figure 5.7. Remote access to SSH Tectia Server with nested tunnels

5.8 Secure System Administration

One of the most widely used applications of Secure-Shell-based products is to replace the unsecured login protocols, Rlogin, Telnet, and FTP with secure alternatives. The SSH Tectia client/server solution is based on SSH Secure Shell, used worldwide for secure system administration.

Figure 5.8 shows a typical SSH Tectia environment for secure system administration. In this example, the managed servers reside in the DMZ network, and the system administrator connects to them over the Internet using SSH Tectia Client. In this scenario, SSH Tectia Client software is installed in the system administrator's workstation and the SSH Tectia Server software in the managed server.

Figure 5.8. Secure system administration with SSH Tectia client/server solution

Unlike the competing Secure-Shell-based products for secure remote administration, SSH Tectia provides also centralized management with SSH Tectia Manager. This eliminates the need for local on-site configuration and maintenance, since SSH Tectia Manager can be used to centrally manage SSH Tectia Client, ConnectSecure, and Server. SSH Tectia Manager helps company IT to reduce the operating costs related to software and configuration management, and offers advanced monitoring capabilities from a central location.

5.9 Secure System Administration with RSA SecurID

One of the benefits of the SSH Tectia client/server solution is its wide support for various user authentication mechanisms, such as RSA SecurID. RSA SecurID is an authentication system that is based on a credit-card-

sized authenticator token that generates a new passcode every 60 seconds. The user combines this code with a secret PIN code that enables secure login to protected resources.

The support for RSA SecurID has been implemented using a method called keyboard-interactive. It enables implementation of new authentication schemes based on keyboard interaction without the need to modify the client side. The keyboard-interactive authentication method is defined in *RFC 4256*.

In this use scenario, SSH Tectia Client is used to perform secure terminal-based administration and file transfer while using RSA SecurID for two-factor authentication. SSH Tectia Server includes support for RSA Authentication Agent API. The RSA Authentication Agent must be installed on the same server. The Agent connects to the RSA Authentication Manager that performs the user identity validation against the one-time password generated by the token.

Figure 5.9 shows the overall architecture of this use scenario.

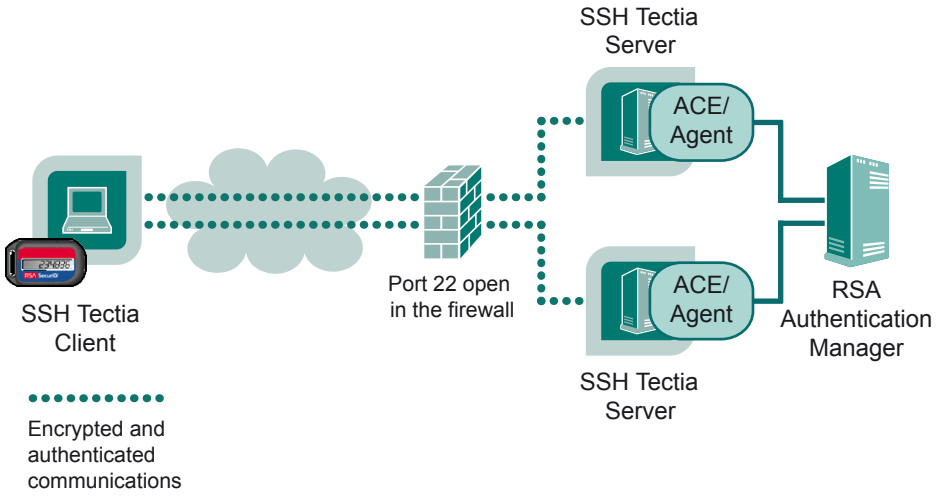

Figure 5.9. Secure system administration using RSA SecurID for authentication

Chapter 6 Product Specification

In this section you can find information on the supported operating systems, hardware requirements and supported authentication methods, cryptographic algorithms and protocols, and third-party products.

6.1 Supported Operating Systems

SSH Tectia client/server solution products can be installed on the following operating systems.

Table 6.1. Supported operating systems per SSH Tectia product

Operating System	Client	ConnectSecure	Server	Server for z/Linux[1]	Server for z/OS[1]
HP-UX (PA-RISC)	11i v1, 11i v2, 11i v3	11i v1, 11i v2, 11i v3	11i v1, 11i v2, 11i v3		
HP-UX (IA-64)	11i v2, 11i v3	11i v2, 11i v3	11i v2, 11i v3		
IBM AIX (POWER)	5.3, 6.1	5.3, 6.1[2]	5.3, 6.1		
Red Hat Enterprise Linux (x86 and x86-64)	4, 5, 5.1, 5.2, 5.3, 5.4	4, 5, 5.1, 5.2, 5.3, 5.4	4, 5, 5.1, 5.2, 5.3, 5.4		
Sun Solaris (SPARC)	9, 10	9, 10	9, 10		
Sun Solaris (x86-64)	10	10	10		
SUSE LINUX Enterprise Desktop (x86 and x86-64)	10	10	10		
SUSE LINUX Enterprise Server (x86)	9, 10	9, 10	9, 10		
SUSE LINUX Enterprise Server (x86-64)	10	10	10		
Microsoft Windows (x86)	XP, Server 2003, Server 2008, Vista, 7	XP, Server 2003	XP, Server 2003, Server 2008		
Microsoft Windows (x64)	Server 2003, Server 2008, Vista, 7		Server 2003, Server 2008		
VMware ESX Server[3]	3.5		3.5		
Red Hat Enterprise Linux for IBM System z (64-bit) [4]				5.1	
SUSE LINUX Enterprise Server for IBM System z (64-bit) [4]				9, 10	
SUSE LINUX Enterprise Server for IBM System z (31-bit)				9	
IBM z/OS (zSeries)					1.8, 1.9, 1.10

[1] Including the client components

[2] SSH Tectia ConnectSecure requires reconfiguring the RBAC mode on AIX 6.1.

[3] SSH Tectia products can be installed directly on the server virtualization layer.

[4] Requires the 31-bit compatibility library `libc.so.6`.

Note

Keep the operating system always fully patched, according to recommendations by the operating system vendor. The minimum patch levels required by SSH Tectia products are mentioned in the SSH Tectia product-specific installation instructions.

6.2 Hardware and Space Requirements

The SSH Tectia products can be run on any standard hardware capable of running the supported operating system versions. The machine should have a TCP/IP connection, and a DVD drive if the software is installed from the installation disk.

The following table summarizes the memory and disk space requirements:

Table 6.2. Memory and disk space requirements

	Client	ConnectSecure	Server	Server for z/Linux	Server for z/OS
RAM	any	any	1 GB[1]	1 GB[1]	1 GB[1]
Disk space	100 MB	100 MB	100 MB	100 MB	200 MB, 250 cylinders

[1] For hundreds of simultaneous tunnels

6.3 SSH Tectia Features per Product

The following table lists the features provided by SSH Tectia Client, ConnectSecure, and Server.

Table 6.3. SSH Tectia products and their features

Feature	Client	Connect-Secure	Server	Server for z/Linux	Server for z/OS
FTP-SFTP conversion		x			x
Transparent FTP tunneling		x			x
Transparent TCP tunneling	x[1]	x			
SFTP API for C (on all platforms)		x			x
SFTP API for Java 1.4.2 (on Linux, Solaris, and Windows)		x			
Checkpoint/restart mechanism		x	x	x	x[2]
Streaming for high-speed transfers		x[3]	x	x	x
Prefix for files in transfer		x	x	x	x
Versatile command-line tools (sshg3, scpg3, sftpg3)	x	x	x[4]	x	x
Terminal GUI on Windows	x	x			
File transfer GUI on Windows	x	x			
Configuration GUI on Windows	x	x	x		
CryptiCore encryption (on Linux and Windows)	x	x	x	x	
Support for connections to standard Secure Shell v2 servers	x	x	x[4]	x	x
Secure terminal server			x	x	x
FTP-SFTP conversion (server side)			x	x	x
SFTP server			x	x	x
Transparent tunneling server			x	x	x
Support for SFTP API			x	x	x
Support for centralized management (requires SSH Tectia Manager)	x	x	x		

[1] Supported as an optional feature on Windows XP.

[2] With HFS files.

[3] Requires SSH Tectia Server as the counterpart.

[4] On Unix platforms (AIX, HP-UX, Solaris, Linux), only. On Windows, SSH Tectia Client needs to be purchased separately.

6.4 Supported Authentication Methods

The following authentication methods and features are supported in the SSH Tectia client/server solution.

6.4.1 Supported User Authentication Methods

The following user authentication methods are supported in the SSH Tectia client/server solution.

Table 6.4. User authentication methods supported by the SSH Tectia client/server solution

Authentication method	SSH Tectia Server			SSH Tectia Client, ConnectSecure, and client tools on Server for z/Linux and Server for IBM z/OS		
	Unix[2]	Windows	z/OS	Unix[2]	Windows	z/OS
Password	x	x	x	x	x	x
Public-key	x	x	x	x	x	x
Certificate	x	x	x[3]	x	x	x[3]
Host-based	x	x	x	x		x
Keyboard-interactive	x	x	x	x	x	x
PAM[1]	x			x	x	x
RSA SecurID[1]	x	x		x	x	x
RADIUS[1]	x	x		x	x	x
GSSAPI/Kerberos	x	x		x	x	

[1] Through keyboard-interactive.

[2] Including SSH Tectia Server for Linux on IBM System z.

[3] Including certificates in files and SAF certificates

6.4.2 Compatibility with OpenSSH Keys

By default, the SSH Tectia client/server solution uses private and public keys stored in the IETF standard Secure Shell v2 format. However, SSH Tectia Client, ConnectSecure, and Server and SSH Tectia Server for Linux on IBM System z can also use keys and related files in the legacy OpenSSH format.

The following OpenSSH-format keys are supported:

- server host key pair
- trusted server host public keys, which clients use to authenticate servers
- user private keys (used by clients to authenticate to a server)
- authorized user public keys (used by a server to authenticate users), including public-key options

6.4.3 Supported PKI Features

- X.509 v3 certificate support

- X.509 v2 CRL fetching via HTTP, LDAP, offline
- OCSP
- SCEP
- PKIX CMPv2 support
- PKCS#1 and PKCS#8 private key support
- PKCS#7 and PKCS#12 import
- MSCAPI support on Windows

6.5 Supported Cryptographic Algorithms, Protocols, and Standards

This section lists the supported cryptographic algorithms and standards supported by SSH Tectia client/server solution.

6.5.1 Public-Key Algorithms

The following public-key algorithms are supported:

- DSA (768-, 1024-, 2048-, or 3072-bit key)
- RSA (768-, 1024-, 2048-, or 3072-bit key)

6.5.2 Data Integrity Algorithms

The following data integrity algorithms are supported:

- CryptiCore (Badger) (16-byte key)
- HMAC MD5 (16-byte key, FIPS PUB 198)
- HMAC SHA-1 (20-byte key, FIPS PUB 198)

6.5.3 Encryption Algorithms

For session encryption, the following symmetric algorithms are supported:

- 3DES (168-bit key)
- AES (128-, 192-, or 256-bit key, CBC or CTR mode)

- Arcfour (128-bit key)
- Blowfish (128-bit key)
- CryptiCore (Rabbit) (128-bit key)
- SEED (128-bit key)
- Twofish (128-, 192-, or 256-bit key)

6.5.4 Hardware Acceleration of Cryptographic Operations

SSH Tectia Server for Linux on IBM System z automatically uses hardware acceleration (CPACF), if it is available, on cryptographic operations with the following algorithms:

- 3DES
- AES
- SHA-1

6.5.5 FIPS-Certified Cryptographic Library

SSH Tectia Client, ConnectSecure, and Server can be operated in FIPS mode, using a version of the cryptographic library that has been certified according to the Federal Information Processing Standard (FIPS) 140-2.

The FIPS 140-2 Cryptographic Library is supported on the following operating systems:

- Microsoft Windows XP, Server 2003, Server 2008, and Vista (for SSH Tectia Client)
- Sun Solaris 9 and 10
- Red Hat Enterprise Linux 4 and 5
- AIX 5.3 and 6.1
- HP-UX 11.11, 11.23 and 11.31

SSH Tectia Server for Linux on IBM System z does not support FIPS-certified cryptographic libraries.

6.6 Supported Third-Party Products

This section lists the third-party products that have been tested to work with the SSH Tectia client/server solution.

6.6.1 Smart Cards/Hardware Tokens (Windows)

- RSA Authentication Manager 5.2 (for SecurID)
- RSA Authentication Manager 6.0 (for SecurID)
- RSA Authentication Manager 6.1 (for SecurID)
- RSA Authentication Agents (for SecurID)
- Secure Computing SafeWord PremierAccess 3.2
- ActivCard ActivClient 5.3 (Supports smart cards from multiple vendors)
- ActivCard Gold 2.2 for CAC (Supports smart cards from multiple vendors)
- Aladdin eToken with eToken RTE v3.60 on supported Windows platforms
- Safenet iKey

6.6.2 Certificate Authorities

- Entrust Authority Security Manager 7.1-7.2
  - Entrust Entelligence Desktop Manager 6.1 and 7.0

    Supported on Microsoft Windows XP.
- Microsoft Windows 2000/2003 Certificate Authority
- RSA Keon

6.6.3 Other Supported Third-Party Products

- RADIUS support with Microsoft IAS and FreeRADIUS
- Active Directory support for SSPI authentication on Windows Server 2003
- Centrify Direct Control 3.0

Open Source Software License Acknowledgements

SSH Communications Security Corp acknowledges the following Open Source Software used in the SSH Tectia client/server solution.

BSD Software

This product includes software developed by the University of California, Berkeley and its contributors.

DES

This product includes software developed by Eric Young `eay@cryptsoft.com`.

PCRE

This software includes PCRE library. Copyright © 1997-2009 University of Cambridge. All rights reserved.

C++ wrapper functions. Copyright © 2007-2009, Google Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the University of Cambridge nor the name of Google Inc. nor the names of their contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS IN-

TERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

XFree86

This Software contains portions of XFree86 software and the delivery of XFree86 software or portions of the said software is subject to the acknowlegement of the following copyright notice and permission notice of The Open Group:

Copyright © 1988, 1998 The Open Group

Permission to use, copy, modify, distribute, and sell XFree86 software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both the copyright notice and this permission notice appear in supporting documentation.

THE XFREE86 SOFTWARE IS PROVIDE "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE OPEN GROUP BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE XFREE86 SOFTWARE OR THE USE OR OTHER DEALINGS IN THE XFREE86 SOFTWARE.

Except as contained in this notice, the name of The Open Group shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from The Open Group.

ZLIB

This software incorporates zlib data compression library by Jean-loup Gailly and Mark Adler.

Glossary

This glossary contains definitions of special terms and abbreviations used in the SSH Tectia user documentation. For more information on terms related to Internet security, see RFC 2828.

Advanced Encryption Standard (AES)
: AES is the current U.S. government standard for a symmetric encryption algorithm. AES is based on the Rijndael *block cipher*. It has a block size of 128 bits and a variable key length of 128, 192, or 256 bits. AES is defined in FIPS 197.

ASCII
: ASCII (American Standard Code for Information Interchange) is an 8-bit character encoding (including a parity bit) commonly used by computers to represent characters of the keyboard.

Arcfour
: Arcfour is a symmetric *stream cipher* with a variable key size. It has been tested to be equivalent of the RC4 cipher by RSA Security.

authentication
: Authentication is the process of verifying that the remote entity is who it claims to be. Authentication includes, for example, verifying that the correct password for the given user account has been entered, but it does not include determining what file system permissions the user has (authorization).

base-64 encoding
: A method of representing six-bit strings of binary data (values 0-63) using 64 ASCII characters. Base-64 encoding was originally used with Privacy Enhanced Mail (PEM), thus it is sometimes referred to as PEM encoding.

block cipher
: A type of symmetric (secret-key) encryption algorithm that encrypts a fixed length block of plaintext (for example, 64 bits) at a time. With a block cipher, the same plaintext block will always encrypt to the same ciphertext block under the same key.

Blowfish
: A symmetric *block cipher*, Blowfish uses a block size of 64 bits and a key length of 32 to 448 bits.

certificate

Certificates are digital documents that are used for verifying the identity of communicating parties. In this documentation, the term certificate is commonly used to refer to *X.509* public-key certificates. A public-key certificate binds identity information about an entity to the entity's public key for a certain validity period.

Certificate Management Protocol (CMP)

CMP defines online interactions between the end entities, the registration authorities, and the certification authority in a PKI. It is defined in RFC 4210.

certificate revocation list (CRL)

CRL is a signed list containing the serial numbers of the certificates that have been revoked or suspended by the certificate issuer (the CA) before their expiration date. The CA usually issues new CRLs at frequent intervals. Current PKIX implementation of CRLs is the *X.509* version 2 CRL. See RFC 3280 for more information.

certification authority (CA)

An entity in a PKI that issues digital certificates (especially *X.509* public-key certificates) and vouches for the binding between the data items in a certificate.

Certificate users (end entities) depend on the validity of information provided by a certificate. Thus, a CA should be someone that the end entities trust, and who usually holds an official position created by and granted power by a government, a corporation, or some other organization.

confidentiality

A security service that protects data from unauthorized disclosure. Usually, unauthorized disclosure of application level data is the primary concern, but the disclosure of the external characteristics of communication can also be a concern in some circumstances. The traffic flow confidentiality service addresses this latter concern by concealing source and destination addresses, message length, or frequency of communication.

Connection Broker

The Connection Broker is a component of SSH Tectia Client, SSH Tectia ConnectSecure, and SSH Tectia MFT Events. It handles all cryptographic operations and authentication-related tasks.

CryptiCore

CryptiCore was introduced in 2005 by the Danish data security company Cryptico. It is a set of algorithms consisting of the *Rabbit* stream cipher, which uses a 128-bit key, and *Badger* data integrity algorithm. CryptiCore enables very fast encryption and integrity checking performance, for example, when used with *Secure Shell*.

Data Encryption Standard (DES)

DES is a U.S. government standard that defines the Data Encryption Algorithm (DEA).

The algorithm itself is a symmetric block cipher with a block size of 64 bits and a key length of 64 bits (of which 8 are parity bits). It was created in the 1970s by IBM, assisted by the U.S. National Security Agency (NSA).

DES is no longer considered secure, but its improved variant **3DES** (also known as TDEA) is still in widespread use. DEA and TDEA are defined in FIPS 46-3.

Diffie-Hellman key exchange

A method for key exchange between two parties. This method can be used to generate an unbiased secret key over an unsecured medium. The method has many variants. A well known attack called the man-in-the-middle attack forces the use of digital signatures or other means of authentication with the Diffie-Hellman protocol.

Digital Signature Algorithm (DSA)

DSA is a digital signature algorithm, invented by the U.S. National Security Agency (NSA). It is defined in the Digital Signature Standard (DSS), FIPS 186-2, alongside with the *SHA-1* hash algorithm.

EBCDIC

EBCDIC (Extended Binary Coded Decimal Interchange Code) is an 8-bit character encoding used by IBM mainframes and other platforms to represent text. EBCDIC was designed in 1964 by IBM and it was the predecessor to *ASCII*. All IBM mainframe operating systems still use EBCDIC.

encryption

A security mechanism used for the transformation of data from an intelligible form (*plaintext*) into an unintelligible form (*ciphertext*), to provide confidentiality. The inverse transformation process is called decryption.

Federal Information Processing Standard (FIPS)

FIPS is a series of U.S. Government technical standards published by the National Institute of Standards and Technology (NIST).

Generic Security Service Application Programming Interface (GSSAPI)

GSSAPI is a function interface that provides security services for applications in a mechanism-independent way. This allows different security mechanisms to be used via one standardized API. GSSAPI is often linked with Kerberos, which is the most common mechanism of GSSAPI. GSSAPI provides support for Windows domain authentication with Active Directory on Windows and Unix. GSSAPI is described in RFC 2743.

hash function

An algorithm that computes a short digest of a longer message. The digest is usually of a fixed size. See also *MD5* and *SHA-1*.

hashed message authentication code (HMAC)

A hashed message authentication code (HMAC) is a type of message authentication code (MAC) calculated using a cryptographic hash function in combination with a secret key. As with any MAC, it can be used to verify both the data integrity and data origin authenticity.

Any iterative cryptographic hash function, such as *MD5* or *SHA-1*, may be used in the calculation of an HMAC. The resulting MAC algorithms are termed HMAC-MD5 or HMAC-SHA-1, respectively. The cryptographic strength of the HMAC depends upon the cryptographic strength of the underlying hash function and on the size and quality of the key.

integrity

A security service that ensures that data modifications are detectable. Although authentication and integrity services are often cited separately, in practice they are intimately connected and almost always offered together.

Keyboard-Interactive

Keyboard-Interactive is a generic authentication method in Secure Shell, used to implement different types of authentication mechanisms. Any authentication method that requires only the user's input can be performed with keyboard-interactive. In SSH Tectia, currently supported methods include password, *PAM*, *RADIUS*, and *RSA SecurID*.

Lightweight Directory Access Protocol (LDAP)

LDAP is a protocol for accessing distributed directory services that support the X.500 directory model. The protocol is especially targeted at management applications and browser applications that provide interactive read/write access to directories. LDAPv3 is defined in RFC 4510.

MD5

A message-digest algorithm that computes an irreversible 128-bit hash value for a document. The algorithm is documented in RFC 1321.

MD5 is no longer considered secure. Other newer algorithms such as *SHA-1* or SHA-256 are recommended instead.

MVS

MVS (Multiple Virtual Storage), first released in 1974, is an operating system used on the IBM mainframe computers. Although the original MVS was discontinued several years ago, the term MVS is still commonly used to refer to operating systems based on the same architecture, including OS/390 and the current z/OS.

MVS systems are traditionally accessed by 3270 terminals, or by PCs running *TN3270* emulators.

Online Certificate Status Protocol (OCSP)

In some applications, such as banking and e-commerce, it may be necessary to obtain certificate revocation status that is more timely than is possible with *CRL*s. OCSP may be used to determine the current revocation status of a digital certificate, instead of or as a supplement to checking against a periodically published CRL. OCSP is described in RFC 2560.

passphrase

A passphrase is a string of characters. Whereas a password is used for authentication directly, a passphrase is only used to protect the actual information used for authentication, the *private key*.

password

A password is a string of characters such as numbers, letters and special characters, used for authenticating an entity against another. The strength of a password is measured by its "randomness", called entropy. If a password has a high level of entropy, it is difficult to guess using dictionary attacks.

Pluggable Authentication Module (PAM)

Pluggable Authentication Module is an authentication framework used in Unix systems. PAM allows stacking authentication modules and can be used to integrate login with different authentication mechanisms.

public-key cryptography

In contrast to symmetric (secret-key) cryptography with just one cipher key, in public-key cryptography each person or host has two keys. One is the *private key*, which is used for signing outgoing messages and decrypting incoming messages, the other is the *public key*, which is used by others to confirm the authenticity of a signed message coming from that person and for encrypting messages addressed to that person. The private key must not be available to anyone but its owner, but the public key is spread via trusted channels to anyone.

Public-Key Cryptography Standards (PKCS)

The PKCS standards are a document series from RSA Laboratories. Some of the most important PKCS standards include:

- PKCS #1 for RSA encryption and signature formats
- PKCS #7 for cryptographic message encapsulation
- PKCS #8 for private-key information syntax
- PKCS #10 for certification requests
- PKCS #11 for a cryptographic token interface commonly used with smart cards
- PKCS #12 for storing or transporting a user's private keys, certificates, and miscellaneous secrets

public-key infrastructure (PKI)

PKI consists of end entities possessing key pairs, *certification authorities*, certificate repositories (directories), and all the other software, components, and entities required when utilizing public-key cryptography.

Remote Authentication Dial-In User Service (RADIUS)

RADIUS is a protocol for checking a user's authentication and authorization information from a remote server. It is originally intended for authenticating dial-in users, but it is also suitable for use with Secure Shell. RADIUS is described in RFC 2865.

Request For Comments (RFC)

A document of the Internet Engineering Task Force (IETF) under standardization. RFCs can be located at the IETF web site at http://www.ietf.org/rfc.html.

Resource Access Control Facility (RACF)

Resource Access Control Facility (RACF) is a security system by IBM that provides access control and auditing functionality for the z/OS operating system.

RSA

RSA is a public-key encryption and digital signature algorithm, invented by Ron Rivest, Adi Shamir, and Leonard Adleman, and defined in PKCS #1. The RSA algorithm was patented by RSA Security, but the patent expired in September 2000.

Secure File Transfer Protocol (SFTP)

The Secure File Transfer Protocol (SFTP) is a de-facto industry standard for securing transfer of files over the network. SFTP is natively supported by the SSH Tectia client/server solution.

SFTP is not technically related to the unsecured File Transfer Protocol (FTP), but the use of SFTP client programs is similar to that of FTP. The server side runs a Secure Shell version 2 server with the SFTP subsystem enabled.

Secure Shell (SecSh)

The Secure Shell (SecSh) protocol was originally developed in 1995 by Tatu Ylönen, the founder of SSH Communications Security. Secure Shell replaces other, unsecured terminal applications (such as Rlogin, Telnet, and FTP), and allows forwarding arbitrary TCP/IP ports over the secure channel, enabling secure connection, for example, to an e-mail service.

There are two versions of the Secure Shell protocol. The current version, Secure Shell version 2 (SecSh v2, SSH2) provides several security improvements as compared to the original Secure Shell version 1 (SecSh v1, SSH1). SSH Tectia is based on SSH2, and SSH Communications Security considers SSH1 deprecated and does not recommend nor support its use anymore. The SSH2 protocol is defined in RFCs 4250-4256.

SecurID

RSA SecurID is a widely-used two-factor authentication method based on the use of SecurID Authenticator tokens. The Authenticator token generates a random numerical code that the user needs to enter when connecting to a system. RSA ACE/Agent is used to verify the code. RSA ACE/Server acts as the management component handling the authentication requests and managing the authentication policies for enterprise networks.

SEED

A strong *block cipher*, SEED uses a block size of 128 bits and a key length of 128 bits. SEED is a national standard encryption algorithm in South Korea. It has also been adopted as an ISO/IEC standard (ISO/IEC 18033-3) and an IETF RFC (RFC 4269).

SHA-1

SHA-1 is an improved version of the original Secure Hash Algorithm (SHA), designed by National Security Agency (NSA). The algorithm produces a 160-bit message digest. It is defined in FIPS 180-1 and it is also part of the Digital Signature Standard (DSS), FIPS 186-2.

SOCKS

SOCKS is a protocol for traversing through application gateway firewalls. It allows an application inside the firewall to access resources on the global Internet. The protocol is defined in RFC 1928.

SSH Tectia client/server solution

The SSH Tectia client/server solution consists of SSH Tectia Client, SSH Tectia ConnectSecure, SSH Tectia Server, SSH Tectia Server for Linux on IBM System z and SSH Tectia Server for IBM z/OS.

SSH Tectia Client

SSH Tectia Client provides secure interactive file transfer and terminal client functionality for remote users and system administrators to access and manage servers running SSH Tectia Server or other applications using the Secure Shell protocol. It also supports (non-transparent) tunneling of TCP-based applications, and on Windows, transparent TCP tunneling.

SSH Tectia ConnectSecure

SSH Tectia ConnectSecure is designed for FTP replacement. It is a client-side product that provides FTP-SFTP Conversion, enhanced file transfer, and transparent FTP and TCP tunneling services for connecting to a Secure Shell server.

SSH Tectia MFT Auditor

SSH Tectia MFT Auditor is a central system for auditing SSH Tectia and OpenSSH file transfers and commands in large environments. It offers tools for statistical analysis, monitoring, planning, and troubleshooting.

SSH Tectia MFT Events

SSH Tectia MFT Events is designed for automating file transfer and command events and for monitoring their performance. SSH Tectia MFT Events is typically installed on a server host and it is capable of connecting to any standard Secure Shell server.

SSH Tectia Manager

SSH Tectia Manager is a security management platform designed to reduce the total cost of ownership of large multi-platform SSH Tectia environments. It enables administrators to enforce consistent security policy and to more efficiently monitor the state of their SSH Tectia security environments.

SSH Tectia Server

SSH Tectia Server is a server-side component where Secure Shell clients connect to. There are three versions of the product available: *SSH Tectia Server* for Linux, Unix and Windows, *SSH Tectia Server for Linux on IBM System z*, and *SSH Tectia Server for IBM z/OS*.

stream cipher

A type of symmetric (secret-key) encryption algorithm that encrypts a single bit at a time. With a stream cipher, the same plaintext bit or byte will encrypt to a different bit or byte every time it is encrypted.

Telnet 3270

TN3270 is a remote-login protocol used by IBM 3270 mainframe computer terminal emulators. Like standard Telnet, TN3270 is natively unsecured.

Transport Layer Security (TLS)

Transport Layer Security is a protocol providing confidentiality, authentication, and integrity for stream-like connections. It is typically used to secure HTTP connections. TLS is defined in RFC 5246.

Twofish

A strong and fast *block cipher*, Twofish was one of the five final candidates for the U.S. Advanced Encryption Standard. Twofish uses a block size of 128 bits and a key length of up to 256 bits.

Unix System Services (USS)

Unix System Services (USS) is a component of the IBM z/OS operating system. It allows Unix applications from other platforms to run on IBM mainframes.

X.509

The ITU-T X.509 recommendation defines the formats for X.509 *certificate* and X.509 *CRL*. Different X.509 applications are further defined by the PKIX Working Group of the IETF. These include X.509 version 3 public-key certificates and X.509 version 2 CRLs.

Index

Symbols

A

B

C

D

E

F

G

H

I

T

U

V

W

X

