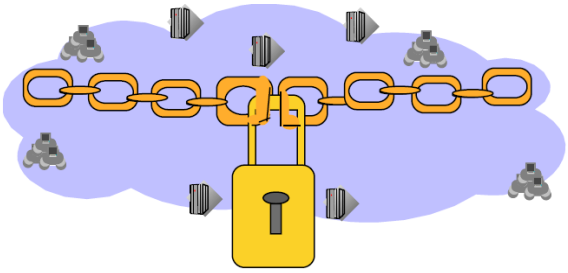
Protocol Comparisons: OpenSSH, SSL/TLS (AT-TLS), IPSec

Author:
- *Gwen Dente, IBM Gaithersburg, MD*



Acknowledgments:
- *Alfred Christensen, IBM*
- *Erin Farr, IBM*
- *Christopher Meyer, IBM*
- *Linwood Overby, IBM*
- *Richard Theiss, IBM*

1. This document was assembled with assistance from information provided by the following people:
   1. Alfred Christensen, IBM
   2. Erin Farr, IBM
   3. Christopher Meyer, IBM
   4. Linwood Overby, IBM
   5. Richard Theiss, IBM

What is OpenSSH

- A secure protocol that provides:
  - Authentication
    - both client and server
  - Data Privacy
    - including Userid/password
  - Data Integrity
  - Authorization – regulates access control to accounts
  - Forwarding (a.k.a. tunneling) – encryption of other TCP/IP-based sessions

Most implementations of OpenSSH support direct access only to UNIX file systems.

Function	OpenSSH Utility	An alternative to…
Secure remote login	ssh, sshd	rlogin, rsh
Secure file transfer	sftp, sftp-server, scp	rcp

OpenSSH additionally provides these utilities:	
Key management	ssh-keygen, ssh-agent, ssh-add, ssh-keyscan

- OpenSSH from IBM z/OS: Program Product: IBM Ported Tools for z/OS
  - Unpriced, runs on z/OS 1.4 and higher
  - Order from ShopzSeries, under “MVS: System Mgmt. and Security.”
  - GA Version info: OpenSSH 3.5p1, OpenSSL 0.9.7b, zlib 1.1.4
  - OA10315 version is: OpenSSH 3.8.1p1, OpenSSL 0.9.7d, zlib 1.1.4
  - Supports SSH V1 and V2

- SSH protects against:
  - Eavesdropping
  - Name service and IP spoofing
  - Connection Hijacking
  - Man-in-the-Middle Attacks
  - Insertion Attacks

1. OpenSSH from IBM z/OS: Program Product: IBM Ported Tools for z/OS
2. `• unpriced, runs on z/OS 1.4 and higher
3. `• order from ShopzSeries, under “MVS: System Mgmt. and Security.”
4. `• GA Version info: OpenSSH 3.5p1, OpenSSL 0.9.7b, zlib 1.1.4
5. `• OA10315 version is: OpenSSH 3.8.1p1, OpenSSL 0.9.7d, zlib 1.1.4
    1. Supports SSH V1 and V2
6. SSH protects against:
    1. Eavesdropping
    2. • Name service and IP spoofing
    3. • Connection Hijacking
    4. • Man-in-the-Middle Attacks
    5. • Insertion Attacks
7. Eavesdropping – network snooper who reads traffic without affecting it.
8. SSH prevents this through encryption
9. Name Service and IP spoofing – an attacker subverts your naming service (DNS, for example) and network programs may connect to the wrong machine.
10. SSH prevents this by cryptographically verifying the server host identity. It may print warning or exit, depending on configuration.
11. Connection Hijacking – an attacker listens to network traffic and injects his own traffic, essentially stealing away the TCP connection from one of its endpoints.
12. SSH prevents this through integrity checking, and shuts down if data was corrupted.
13. Man-in-the-middle attack – an attacker is able to read, and possibly modify, transmissions between two parties/hosts. Neither party knows they have been attacked. Each believes they are still communicating with his intended destination.
14. SSH prevents this 2 ways:
15. 1. server host authentication
16. 2. allows users to limit the authentication methods vulnerable to this attack (like password authentication). Public-key and Hostbased/RhostsRSA are immune.
17. Insertion Attack – an attacker inserts arbitrary data into the data stream
18. SSH (Protocol Version 2) uses cryptographically strong integrity checks to avoid this.
19. Base SSH: Uses Public Key Infrastructure for authentication and encryption,
20. Authentication (both client and server) through:
21. – Public key cryptography
22. – Existing login passwords
23. – Trusted hosts authentication
24. • Data Privacy - through encryption
25. • Data Integrity - guarantees data traveling over the network is unaltered
26. • Authorization – regulates access control to accounts
27. • Forwarding (a.k.a. tunneling) – encryption of other TCP/IP-based sessions
28. SSH does not protect against:
29. Password cracking – an attacker steals your password
30. SSH alleviates this by:
31. 1. Encrypting passwords as they pass over the network.
32. 2. Providing public-key authentication. Stolen passphrase is useless without the private key file.
33. IP and TCP attacks – SSH operates on top of TCP, so it is vulnerable to attacks against weaknesses in TCP and IP.
34. SSH, through privacy, integrity, and authentication, minimizes these attacks to denial-of-service attacks.
35. Traffic analysis - Deducing useful intelligence from patterns of message traffic, without breaking codes or reading the messages
36. SSH does not address traffic analysis attacks.
37. Covert Channel – a hidden communication medium, usually defying the host’s security policy.

Comparison: SSL/TLS or AT-TLS, IPSec, & OpenSSH

Public Key Technology	Security: • Authentication of Partner • Data Integrity Checking • Encryption of Userid, Password, Data	IP Protocol Protected?	Types of Files?	Number of "sessions" on encrypted pipe
SSL/TLS or AT-TLS with x.509 Certificates	Yes (server authentication required) • Server Authentication with Server Certificate • Optional Client Authentication with Client Certificate	Protects TCP	MVS datasets and UNIX files	1 per TCP connection
IPSec with x.509 Certificates	Yes (mutual authentication required) • Partner Authentication required with Certificate at both endpoints.	Protects TCP, UDP, any IP protocol	MVS datasets and UNIX files	Multiple per connection
OpenSSH with Public and Private Key Pair (Assumption: SSH V2)	Yes (mutual authentication required) • Server Authentication with Server Public/Private Key Pair – Server Public Key has been loaded into Client-Side "$HOME/.ssh/known_hosts" file and • Client Authentication with Client password or with Public/Private Key Pair – Client Public Key has been loaded into Server-Side "$HOME/.ssh/authorized_keys" file and	Protects TCP	UNIX files only**	**No SSH Tunnel:** 1 per TCP connection **SSH Tunnel:** Multiple per TCP connection

**** Note:** OpenSSH works with UNIX file systems only; EOM vendor implementations exist to work with MVS files. OpenSSH can operate against MVS files if they have been copied or moved into an HFS or zFS.

1. OpenSSH from IBM z/OS: Program Product: IBM Ported Tools for z/OS
2. `•` unpriced, runs on z/OS 1.4 and higher
3. `• order from ShopzSeries, under “MVS: System Mgmt. and Security.”
4. `• GA Version info: OpenSSH 3.5p1, OpenSSL 0.9.7b, zlib 1.1.4
5. `• OA10315 version is: OpenSSH 3.8.1p1, OpenSSL 0.9.7d, zlib 1.1.4
6. Base SSH: Uses Public Key Infrastructure for authentication and encryption,
7. Authentication (both client and server) through:
8. – Public key cryptography
9. – Existing login passwords
10. – Trusted hosts authentication
11. • Data Privacy - through encryption
12. • Data Integrity - guarantees data traveling over the network is unaltered
13. • Authorization – regulates access control to accounts
14. • Forwarding (a.k.a. tunneling) – encryption of other TCP/IP-based sessions
15. BUT key management and distribution for large numbers of users are difficult because there is no concept of a Certificate Authority. As a result, the keys themselves or the trusted hosts file itself for each server needs to be distributed to the participants. In addition, the SSH option for eliminating
16. Only for UNIX files with SFTP; only for UNIX shell with SSH (Tectia extensions allow usage on MVS files); only uses crypto card for generation of the keys
17.
18. Base SSL or TLS: Either Unix or MVS files with either TN3270 or FTPS;
19. also uses Public Keys, but in addition relies on x.509 Certificates for authentication thus simplifying key management and distribution, especially if you use a well-known Certificate Authority;
20. can store master key in hardware and can take advantage of crypto cards for handshakes and sometimes data encryption

Appendix A: Protocol Comparisons -- IPSec and SSL; OpenSSH

1. The material in this appendix is repeated in the IPSec Overview, another presentation in this course.

IPSec vs. AT-TLS

	IPSec	AT-TLS
Traffic protected with data authentication and encryption	All protocols	TCP
End-to-end protection	Yes	Yes
Segment protection	Yes	No
Scope of protection	Security association 1)all traffic 2)protocol 3)single connection 4)Certificate Content is protected	TLS session 1)single connection 2)Certificate Content flows in clear
How controlled	IPSec policy 1)z/OS responds to IKE peer 2)z/OS initiates to IKE peer based on outbound packet, IPSec command, or policy autoactivation	AT-TLS policy 1)For handshake role of server, responds to TLS client based on policy 2)For handshake role of client, initializes TLS based on policy 3)Advanced function applications
Requires application modifications	No	No, unless advanced function needed 1)Obtain client cert/userid 2)Start TLS
Type of security	Device to device	Application to application
Type of authentication	Peer-to-peer	1)Server to client 2)Client to server (opt)
Authentication credentials	1)Preshared keys 2)X.509 certificates	X.509 certificates
Authentication principals	Represents host	Represents user
Session key generation/refresh	"Yes" with IKE "No" with manual IPSec	TLS handshake

1. This chart was assembled by Alfred Christensen, Linwood Overby, and Christopher Meyer of IBM Raleigh CommServer Development.
2. There is one more difference between the two protocols that is not depicted on this chart:
   1. If the SSL Private key is compromised, then any negotiation of a new Session Key is compromised
   2. If the IPSec Private key is compromised, then there is a bit more protection, because the Session Key is independently negotiated with Diffie-Hellman.

Comparison: SSL/TLS or AT-TLS, IPSec, & OpenSSH

Visual duplicated in order to provide context for remaining visuals.

Public Key Technology	Security: • Authentication of Partner • Data Integrity Checking • Encryption of Userid, Password, Data	IP Protocol Protected?	Types of Files?	Number of "sessions" on encrypted pipe
SSL/TLS or AT-TLS with x.509 Certificates	Yes (server authentication required) • Server Authentication with Server Certificate • Optional Client Authentication with Client Certificate	Protects TCP	MVS datasets and UNIX files	1 per TCP connection
IPSec with x.509 Certificates	Yes (mutual authentication required) • Partner Authentication required with Certificate at both endpoints.	Protects TCP, UDP, any IP protocol	MVS datasets and UNIX files	Multiple per connection
OpenSSH with Public and Private Key Pair (Assumption: SSH V2)	Yes (mutual authentication required) • Server Authentication with Server Public/Private Key Pair – Server Public Key has been loaded into Client-Side "$HOME/.ssh/known_hosts" file and • Client Authentication with Client password or with Public/Private Key Pair – Client Public Key has been loaded into Server-Side "$HOME/.ssh/authorized_keys" file and	Protects TCP	UNIX files only**	**No SSH Tunnel:** 1 per TCP connection **SSH Tunnel:** Multiple per TCP connection

**** Note:** OpenSSH works with UNIX file systems only; EOM vendor implementations exist to work with MVS files. OpenSSH can operate against MVS files if they have been copied or moved into an HFS or zFS.

1. This is a duplicate of an earlier page and is reproduced to put the other pages of this appendix in context.
2. OpenSSH from IBM z/OS: Program Product: IBM Ported Tools for z/OS
3. `•` unpriced, runs on z/OS 1.4 and higher
4. `• order from ShopzSeries, under “MVS: System Mgmt. and Security.”
5. `• GA Version info: OpenSSH 3.5p1, OpenSSL 0.9.7b, zlib 1.1.4
6. `• OA10315 version is: OpenSSH 3.8.1p1, OpenSSL 0.9.7d, zlib 1.1.4
7. Base SSH: Uses Public Key Infrastructure for authentication and encryption,
8. Authentication (both client and server) through:
9. – Public key cryptography
10. – Existing login passwords
11. – Trusted hosts authentication
12. • Data Privacy - through encryption
13. • Data Integrity - guarantees data traveling over the network is unaltered
14. • Authorization – regulates access control to accounts
15. • Forwarding (a.k.a. tunneling) – encryption of other TCP/IP-based sessions
16. BUT key management and distribution for large numbers of users are difficult because there is no concept of a Certificate Authority. As a result, the keys themselves or the trusted hosts file itself for each server needs to be distributed to the participants. In addition, the SSH option for eliminating
17. Only for UNIX files with SFTP; only for UNIX shell with SSH (Tectia extensions allow usage on MVS files); only uses crypto card for generation of the keys
18.
19. Base SSL or TLS: Either Unix or MVS files with either TN3270 or FTPS;
20. also uses Public Keys, but in addition relies on x.509 Certificates for authentication thus simplifying key management and distribution, especially if you use a well-known Certificate Authority;
21. can store master key in hardware and can take advantage of crypto cards for handshakes and sometimes data encryption

Comparison: SSL/TLS or AT-TLS, IPSec, & OpenSSH

Public Key Technology	Private Key Repository	Key Management	Encryption Protocols	Hashing Protocols
SSL/TLS or AT-TLS with x.509 Certificates	In Keyring or key database	Usually low maintenance: Certificate Authority Services for x.509 cert.***	Asymmetric: RSA for Server (and optionally Client) Authentication and Generation of Session Key Symmetric: RC2, RC4, DES, 3DES, AES128, AES256	MD5, SHA-1
IPSec with x.509 Certificates	In Keyring or key database	Usually low maintenance: Certificate Authority Services for x.509 cert.***	Asymmetric: RSA for peer authentication; Diffie-Hellman (DH) for Generation of Session Key Symmetric: RC2, RC4, DES, 3DES, AES128, AES256	MD5, SHA-1
OpenSSH with Public and Private Key Pair (Assumption: SSH V2)	In a trusted hosts file or authorized keys file*	Can be high maintenance: Distribution of each public key; verification at server***	Asymmetric: RSA, DSA for peer authentication and Generation of Session Key and Generation of Digital Signature Symmetric: DES, 3DES, AES128, AES192, AES256	MD5, SHA-1, RIPEMD-160

*** Note:** The system-wide known hosts file is in /etc/ssh/ssh_known_hosts.
The user-specific file is in $HOME/.ssh/known_hosts (Server ID) or $HOME/.ssh/authorized_keys (Client ID)

*****Note:** With x.509, each client participant needs a copy of only the Certificate Authority's certificate that signed the server certificate in its keyring or key database in order to authenticate the server.

For example, 20 server certificates may have been signed by the same CA, and yet the client requires only the 1 trusted CA certificate in order to authenticate the server(s). With OpenSSH each client requires a copy of the public key for each server in order to authenticate that server. If there are 20 servers, then the client requires copies of 20 public keys.

1. Diffie-Hellman key exchange (D-H) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.
2.
3.
4. Perform setup for server authentication:
    1. – Generate host keys for server
    2. • allows a client to verify the identity of the server.
5. • Use ssh-keygen to create host keys:
    1. ssh keygen t dsa f /etc/ssh/ssh_host_dsa_key N ""
    2. ssh keygen t rsa f /etc/ssh/ssh_host_rsa_key N ""
6. – Create local and remote ssh_known_hosts files
    1. • Contains host public keys for all hosts you know about
    2. • Copy local host's public keys to the remote hosts
    3. • Gather public keys of remote hosts
7.
8. OpenSSH from IBM z/OS: Program Product: IBM Ported Tools for z/OS
9. `•` unpriced, runs on z/OS 1.4 and higher
10. `• order from ShopzSeries, under "MVS: System Mgmt. and Security."
11. `• GA Version info: OpenSSH 3.5p1, OpenSSL 0.9.7b, zlib 1.1.4
12. `• OA10315 version is: OpenSSH 3.8.1p1, OpenSSL 0.9.7d, zlib 1.1.4
13. Base SSH: Uses Public Key Infrastructure for authentication and encryption,
14. Authentication (both client and server) through:
15. – Public key cryptography
16. – Existing login passwords
17. – Trusted hosts authentication
18. • Data Privacy - through encryption
19. • Data Integrity - guarantees data traveling over the network is unaltered
20. • Authorization – regulates access control to accounts
21. • Forwarding (a.k.a. tunneling) – encryption of other TCP/IP-based sessions
22. BUT key management and distribution for large numbers of users are difficult because there is no concept of a Certificate Authority. As a result, the keys themselves or the trusted hosts file itself for each server needs to be distributed to the participants.
23. Only for UNIX files with SFTP; only for UNIX shell with SSH (Tectia extensions allow usage on MVS files); only uses crypto card for generation of the keys
24.
25. Base SSL or TLS: Either Unix or MVS files with either TN3270 or FTPS;
26. also uses Public Keys, but in addition relies on x.509 Certificates for authentication thus simplifying key management and distribution, especially if you use a well-known Certificate Authority;
27. can store master key in hardware and can take advantage of crypto cards for handshakes and sometimes data encryption
28.
29.

Appendix B:
Use of Cryptographic Hardware: SSL/TLS, IPSec, SSH

Performance Reports for z/OS Communications Server and Security:

http://www-01.ibm.com/support/docview.wss?uid=swg27005524

1. Some of this material is a subset of the material that is available in the presentation built by Christopher Meyer, of z/OS Communications Server Development in Raleigh, North Carolina.

Use of Cryptographic Hardware: SSL/TLS, AT-TLS

Algorithm	*Available Hardware on z Platform*	
	CPACF only	CPACF + Coprocessor/Accelerator
RSA signature generation	In software	In coprocessor mode only. Otherwise in software (accelerator does not support this operation).
RSA signature verification	In software	In coprocessor/accelerator.
PKA encrypt/decrypt for handshake	In software	In coprocessor/accelerator
SHA-1 digest generation	CPACF	
SHA-224 digest generation	CPACF	
SHA-256 digest generation	CPACF	
SHA-384 digest generation	In software on z9, CPACF in z10 EC	
SHA-512 digest generation	In software on z9, CPACF in z10 EC	
DES encrypt/decrypt	CPACF	
3DES encrypt/decrypt	CPACF	
AES-128 encrypt/decrypt	CPACF	
AES-256 encrypt/decrypt	In software on z9, CPACF in z10 EC	

Extracted from a presentation by Christopher Meyer, IBM Raleigh Communications Server Development Team: "Crypto Demystified"

1. Starting with the z10 processor, more hashing and encryption algorithms are processed in CPACF than were processed on previous System z models: SHA-384, SHA-512, AES-256.

Use of Cryptographic Hardware: IKE

- **Diffie-Hellman based symmetric key generation**
  - Generated keys are used to encrypt traffic that flows over Phase 1 SA
- **RSA signature generate, signature verify for peer authentication**
  - Due to z/OS IKED single-threaded design, multiple Coprocessors or Accelerators will not provide any significant advantage for IKE operations
- **DES, 3DES, AES encryption of IKE payloads**
  - AES requires ICSF (unsupported if ICSF is not available)
- **SHA-1, MD5 HMACs for message authentication**

	Available Hardware on z Platform	
Algorithm	**CPACF only**	**CPACF + Coprocessor/Accelerator**
Diffie-Hellman operations	In software via System SSL	In software via System SSL
RSA signature generation (clear key only)	In software via System SSL	In Coprocessor (not accelerator) if available, otherwise in software via System SSL
RSA signature verification	In software via System SSL	In Coprocessor/Accelerator
DES	In software	
3DES	In software	
AES-128	In CPACF via ICSF, otherwise not supported	
SHA-1	In software	
MD5	In software	

Extracted from a presentation by Christopher Meyer, IBM Raleigh Communications Server Development Team: "Crypto Demystified"

1. Diffie-Hellman key exchange (D-H) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.

Use of Cryptographic Hardware: IPSec

- **DES, 3DES, AES encryption of IKE payloads**
  - AES requires ICSF (unsupported if ICSF is not available)
- **SHA-1, MD5 HMACs for message authentication**

	Available Hardware on z Platform
Algorithm	**CPACF** (stack doesn't use coproc'r or accel'r)
DES	In CPACF (via ICSF)
3DES	In CPACF
AES-128	In CPACF
SHA-1	In CPACF (via ICSF)
MD5	In software

- **zIIP Offload**
  - Starting with V1R8 (APAR PK40178), all SRB-based processing in stack, ***including these crypto operations***, can be offloaded to zIIP to reduce cost of IPSec protection.
    - IPSec processing that is based on enclave SRBs is eligible for zIIP offload.
      - Only an outbound-initiated IPSec operation is ineligible as it is processed on a TCB. (See notes.)

Extracted from a presentation by Christopher Meyer, IBM Raleigh Communications Server Development Team: "Crypto Demystified"

1. The zIIP assisted IPSec function is designed to move most of the IPSec processing from the general purpose processors to the zIIPs.
2. z/OS CS TCP/IP recognizes IPSec packets and routes a portion of them to an independent enclave SRB – this workload is eligible for the zIIP.
   1. Inbound operation (not initiated by z/OS)
      1. All inbound IPSec processing is dispatched to enclave SRBs and is eligible for zIIP
      2. All subsequent outbound IPSec responses from z/OS are dispatched to enclave SRB. This means that all encryption/decryption of message integrity and IPSec header processing is sent to zIIP
   2. Outbound operation (initiated by z/OS)
      1. Operation which starts on a TCB is not zIIP eligible
      2. BUT… any inbound response or acknowledgement is SRB-based and therefore zIIP eligible
      3. AND… all subsequent outbound IPSec responses from z/OS are also zIIP eligible

Use of Cryptographic Hardware: OpenSSH

	Available Hardware on z Platform	
Cipher	**CPACF only**	**CPACF + Coprocessor**
RNG (Random Number Generation)	In software	In Coprocessor
RSA	In software	In software
DSA	In software	In software

- RSA, DSA
  - for Authentication of peer
  - for Generation of Session Key and Digital Signature
- RNG
  - OpenSSH on z accesses the hardware Coprocessor only during the Random Number Generation (RNG) that is used in the process of generating the symmetric key which will be used during the data transfer stage of SSH.

1. RSA is the default. Choose DSA only when you have a specific need for DSA.
2. Diffie-Hellman key exchange (D-H) is a cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher.
3. As documented at z/OS IBM Ported Tools for z/OS User's Guide (SA22-7985-04), on page 27, OpenSSH uses hardware support to generate random numbers through CSFRNG (random number generate service). CSFRNG is a system z nomenclature to CSNBRNG.
   1. According to the z/OS Cryptographic Services ICSF Application Programmer's Guide (SA22-7522-09) on page 167 you will find the CSNBRNG description. On page 168, under the CSNBRNG Usage Notes, you will find the required hardware to run this callable service.
      1. At IBM z900 / z800, the CCF (Cryptographic Coprocessor Facility) is enough.
      2. At IBM z9 BC / EC, the CEX2C (Crypto Express 2 Card) is required.
      3. In order to exploit the CPACF, OpenSSH developer should code the KMC (Cipher Message with Chaining) with PRNG instruction instead of calling ICSF. (Documented at z/Architecture Principles of Operation (SA22-7832-05) on page 7-47.)
         1. As of 2008 there are no plans for OpenSSH to exploit CPACF -- that is, to switch the ICSF call to the KMC-PRNG instruction.

End of Topic

End of Topic