Comparing Traditional FTP and Secure FTP Over OpenSSH on z/OS Communications Server

Brittany Barton
brittany.barton@ibm.com
Technical Enablement Specialist
Worldwide System Center

Session: TECH_175

Agenda

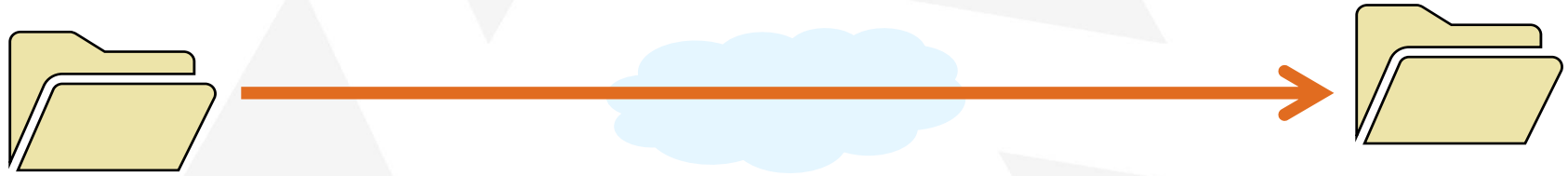
1. Agenda
2. Securing and Protecting Data
3. Terminology
4. FTP Overview
5. Securing FTP with TLS on z/OS
6. Securing FTP with Open SSH on z/OS

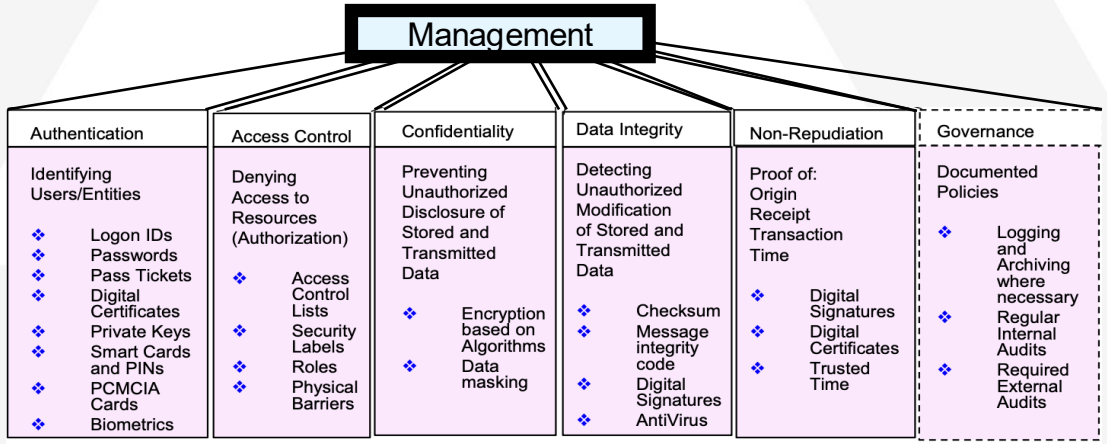
Securing and Protecting Data

Protection and Security Services for Transferring Files

Objectives

1. Authenticate the sender and/or recipient.
2. Keep the data confidential through encryption or masking.
3. Verify the data has not been changed in flight.
4. Verify that the sender is the legitimate owner of the identity he has assumed.





Terminology

Acronyms

Acronym	Meaning	RFC	Function
SFTP	Simple File Transfer Protocol	913	File Transfer - TCP Port 115
TFTP	Trivial File Transfer Protocol	1350	File Transfer - UDP Port 69
SCP	Secure Copy Program	BSD Remote Copy Protocol	Secure File Transfer over SSH - TCP Port 22
FTP	File Transfer Protocol	959 & 2428	File Transfer - TCP Ports 20 & 21
SFTP	SSL File Transfer Protocol	959, 2428 & 4217	Secure File Transfer – TCP Ports 20 & 21
FTPS	File Transfer Protocol Secure or File Transfer Protocol SSL	959, 2228 & 4217	Secure File Transfer – TCP Ports 20 & 21
ftpd & ftpdns	file transfer protocol daemon & file transfer protocol new server	959, 2228 & 4217	File Transfer Unix processes for FTP Server and forked task for client log in – TCP Ports 20 & 21
SFTP & FTP over SSH	SSH File Transfer Protocol or File Transfer Protocol over SSH	959, 2428 & 4251	Secure File Transfer - TCP Port 22
SSH	Secured Shell	4251	Secure Tunnel for communication
sftp & sftpd	secure file transfer protocol client & daemon	959, 2428 & 4251	Secure File Transfer server and listener - TCP Port 22
MFTP	Managed File Transfer Protocol	Proprietary	File Transfer with automated recovery

Terminology

FTP

(File Transfer Protocol)

- RFC959 FTP
- Protocol that most people know and are familiar with
- Extended several times since original release in 1985
- RFC959 Client talks to RFC959 server
- Supported by z/OS Communication Server for many years

FTPS

(File Transfer Protocol Secure)

- RFC4217 FTP over TLS
- RFC4217 Client talks to RFC4217 server which are both extended to support TLS
- Supported by z/OS Communication Server for many years

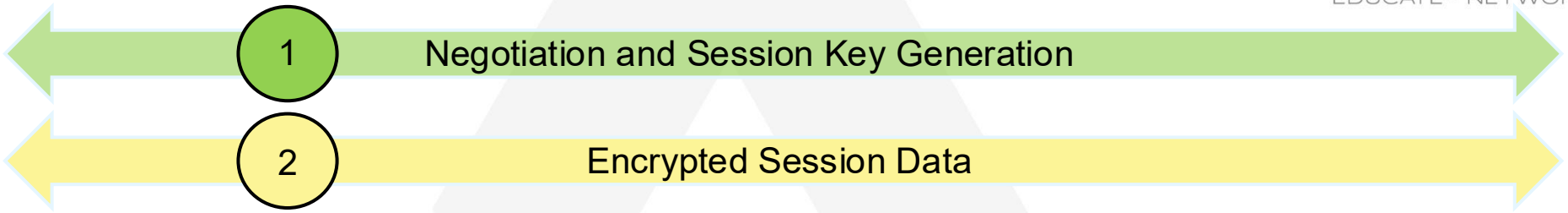
sftp

(Secure Shell File Transfer Protocol)

- Sub-protocol of ssh
- Supported on z/OS version 1.5 and later
- Incompatible with RFC959 – no correlation
- sftp client talks to sftp server – no RFC959 server involved

Overview FTP

General Architecture Encryption Flow



Encryption Flow	What Happens	SSL/TLS (ATTLS)	Open SSH
Stage 1: Asymmetric Algorithms	Negotiation of Secure Connection: Authentication and Generation of and encrypted transmission of Session Key	Handshake Layer	Negotiation Stage
Stage 2: Symmetric Algorithms	Encryption and Decryption of Data Payload (Session Data)	Record Layer	Data Transfer Stage

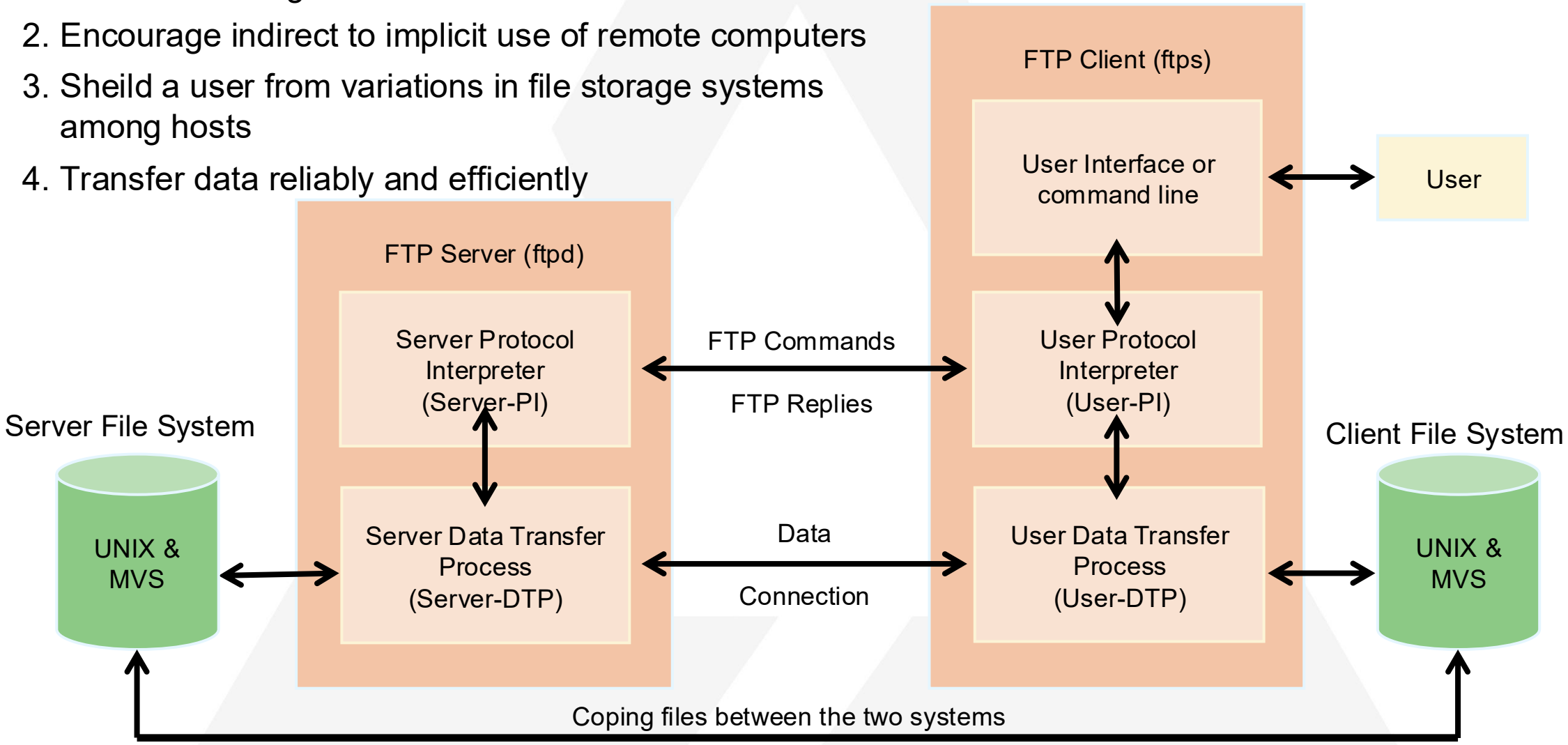
Basically, protocols are using the same basic architecture

1. Authenticate the partner; generate symmetric key
   a) Encrypt symmetric key with asymmetric algorithm and send
2. Encrypt session data with symmetric (“session”) key and transmit the session data

File Transfer Protocol (RFC959)

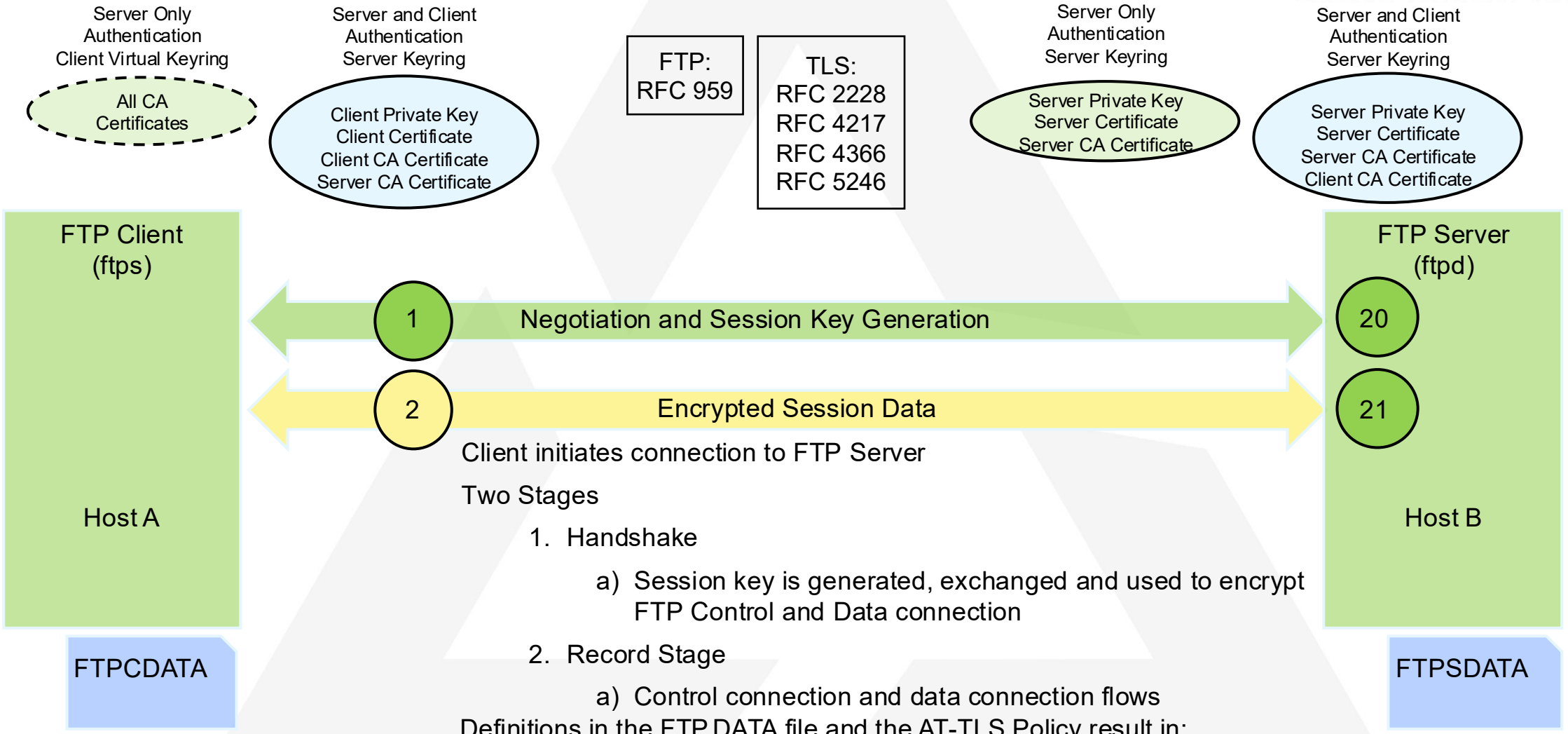
Objective

1. Promote sharing of files
2. Encourage indirect to implicit use of remote computers
3. Sheild a user from variations in file storage systems among hosts
4. Transfer data reliably and efficiently



Securing FTP with TLS on z/OS

Secure FTP with TLS



Client initiates connection to FTP Server

Two Stages

1. Handshake
   a) Session key is generated, exchanged and used to encrypt FTP Control and Data connection
2. Record Stage
   a) Control connection and data connection flows

Definitions in the FTP.DATA file and the AT-TLS Policy result in:

1. Server authentication
2. Client authentication is optional

Securing with TLS

1. Flexible Security Levels
   - Confidentiality, integrity, and authentication
   - Clients/servers ability to dynamically decide on the level of security requires for data transfers
2. Strong authentication
   - Ensuring FTP Client is connected to legitimate destination
3. Identity Management
   - Use TLS identities to authenticate client users and their host machines
4. Public key agreements
   - Use of well-established client identity mechanisms during authentication phase, certificate management may be built into ventral function
   - Offered advantages in certain structured environments

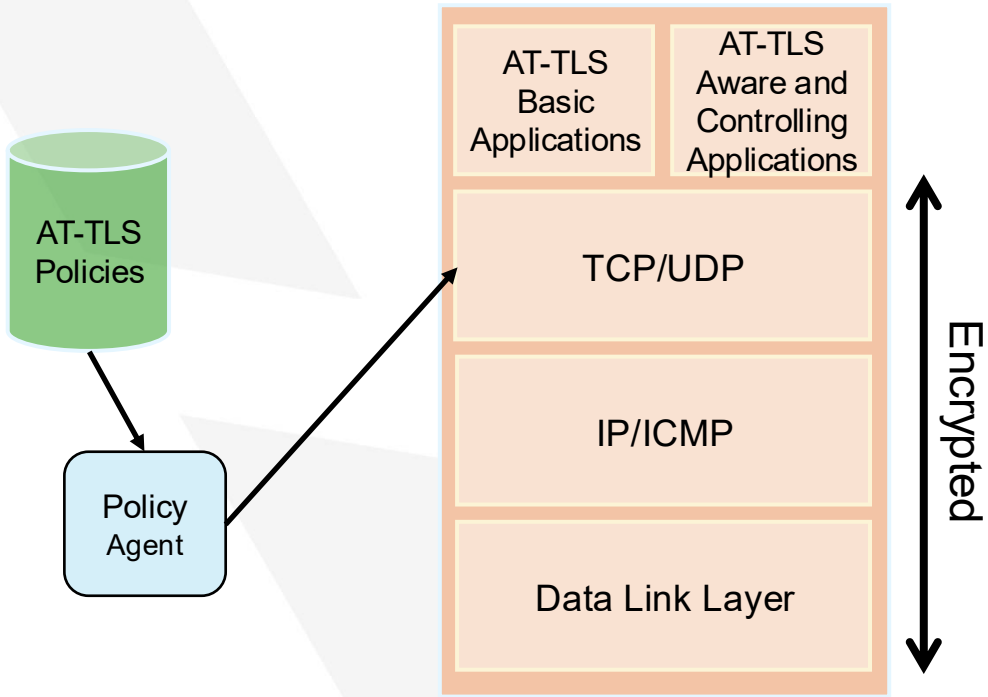
CA Certificate and Key Pair

Server Certificate and Key Pair

Application Transparent – Transport Layer Security AT-TLS

AT-TLS invokes System SSL TLS processing at the TCP layer for the application

- AT-TLS controlled through policy
  - Installed through policy agent
  - Configured through Configuration Assistant GUI or by manual edit of policy files
- AT-TLS Basic applications
  - For Server Only Authentication or Server with “plain” Client Authentication there is no application change required.
- AT-TLS Aware applications
  - Applications can optionally exploit advanced features using SIOCTTLSCTL ioctl call.
  - Required for Client Authentication Advanced Features.
  - Extract information (policy, handshake results, x.509 client certificate, userid associated with certificate)
- AT-TLS Controlling applications
  - Required for a single port to concurrently connect to unsecure clients and secure clients
  - Control if/when to start/stop TLS, reset session/cipher



Negotiation on Control Port

Client

- RFC2228
- Recommended: send AUTH command with the parameter requesting TLS {TLS-PARM} (‘TLS’)
- Needs to behave according to policies depending on the response received from the server and the result of the TLS negotiation
- If client receives AUTH as rejected, client can proceed with the session unprotected **(generic)**

Server

- Sever not allowed to dictate client behavior **(depends on how it is set up),** however, sever can refuse to accept certain FTP commands until the session is secured to a level that is acceptable to the server
- ‘234’ s the server response to an ‘AUTH TLS’ command that it will honor

Clearing the Control Port

1. Protect control port during only part of the session and then revert to plaintext connection

   Due to limitations of boundary devices (NAT and firewalls) which expect to be able to examine the content of the control connection to modify behavior

2. AUTH, USER, PASS, PBSZ and PROT commands

   Protected within TLS protocol

   Then CCC command would be issued to return plaintext socket state

   •CCC Command Behaviors

- **500 Reply**
  Server does not accept/understand the CCC command
- **533 Reply**
  Control Connection is not protected with TLS
- **534 Reply**
  Server does not wish to allow control connection
- **200 Reply**
  Server is accepting the connection
  Shutdown TLS session on the socket and leave it open
  Continue control connection in plaintext, expecting the next command from the client to be in plaintext
  Do not accept any more PBSZ or PROT commands
  All subsequent data transfers must be protected with the current PROT settings

Data Connection Behavior

Classic FTP client server data exchange

- Client obtains port
- Sends the port number to the server
- Sever connects to the client
- The client issues a send or receive request to the server on the control connection and the data transfer commences on the data connection

Firewall-Friendly client/server data exchange

- RFC1579: using the PASV command to reverse the direction of the data connection
- Client requests server open a port
- Server obtains a port and returns the address and port number to the client
- Client connects to the server on this port
- Client issues a send to receive request on the control connection and the data transfer commences on the data connection

Client-initiated server/server data exchange (proxy or PASV connection)

- Client requests server A opens a port
- Server A obtains a port and returns it to the client
- Client sends this port number to server B
- Server B connects to Server A
- Client sends a send or receive request to server A and the complement to server B and the data transfer commences
- Note: Server A is the proxy or PASV and is a client for the Data Connection to server B

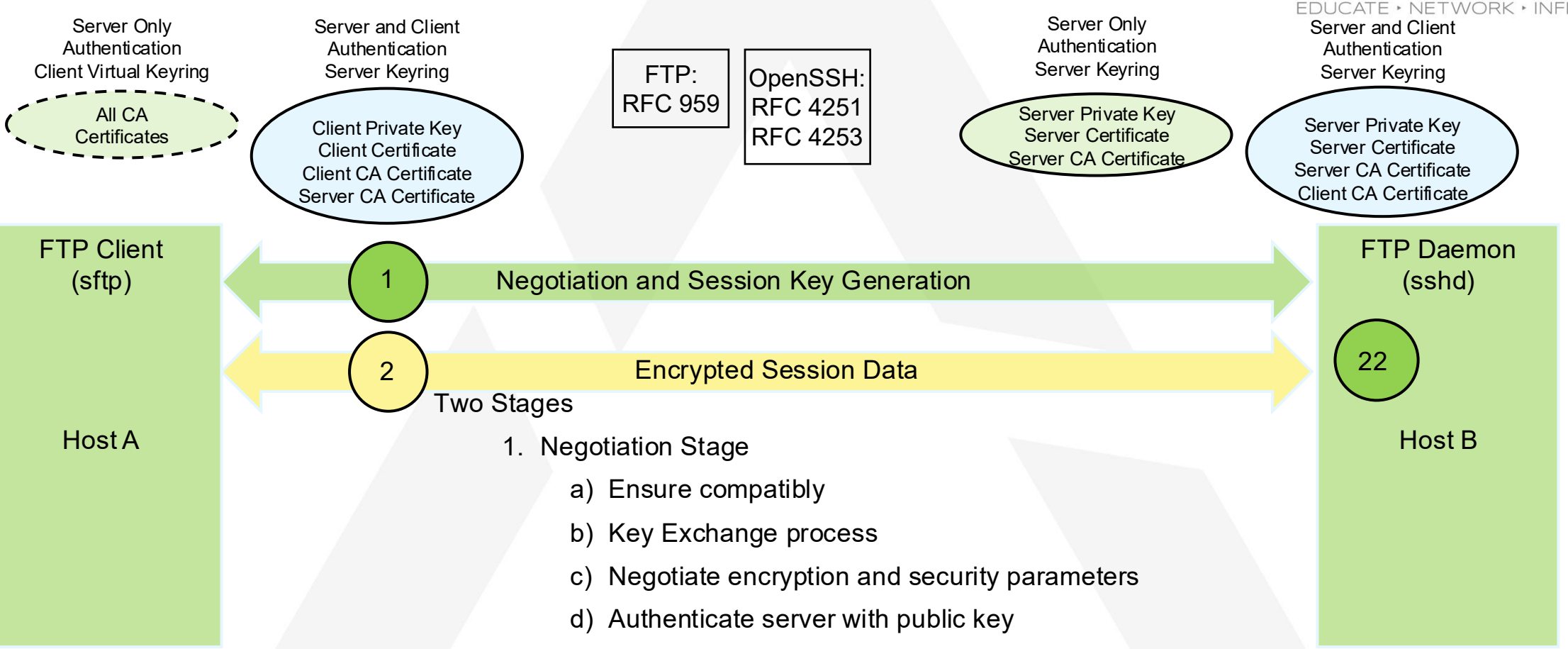
Note: For classic FTP and Firewall FTP, the FTP client MUST be the TLS client and the FTP server MUST be the TLS server

FTPS Protection Mechanisms

- **Secure Initiation:**
  - Client requests encryption using the **AUTH TLS** command.
  - Server may block sensitive commands (e.g., USER, PASS) until the TLS handshake is complete.'
- **Layered Channel Protection:**
  - **Control Channel:** Secured immediately after AUTH TLS to protect credentials and commands.
  - **Data Channel:** Requires the **PROT P** (Private) command to encrypt actual file transfers.
- **The "PBSZ" Requirement:**
  - Clients must issue **PBSZ 0** (Protection Buffer Size) before the PROT command to satisfy TLS protocol standards.
- **Enforcement & Error Handling:**
  - Servers configured for high security will reject STOR (upload) or RETR (download) attempts with a **522 error** if the data channel remains unencrypted.
- **Cipher Suite Negotiation:**
  - Protection strength (e.g., AES-256) is determined by the underlying **TLS handshake**, not the FTP application itself.

Securing FTP with Open SSH

Secure FTP with Open SSH



Two Stages

1. Negotiation Stage
   a) Ensure compatibly
   b) Key Exchange process
   c) Negotiate encryption and security parameters
   d) Authenticate server with public key
   e) Keys exchanged and secure channel established
2. Data Transfer Stage
   a) Encrypt Data with shared secret key
   b) Message Authentication Code (MAC) created as a digital seal

OpenSSH does not use X.509 certificates but does use Public / Private key pairs.

Process

1. **Start the sftp server**:
    a) Ensure that the z/OS sftp server is configured and active
    b) Might involve starting the necessary started tasks or procedures.
2. **Connect to the z/OS sftp server**
    a) Use an sftp client on your local machine.
    b) Can connect using the following syntax in your terminal:
        a) sftp -o BatchMode=no [username]@[z/OS_host]
            a) [username] = your z/OS user ID and
            b) [z/OS_host] = the hostname or IP address of your z/OS system.
3. **Navigate and Transfer Files**:
    a) Navigate to the z/OS file system and transfer files to and from your local machine using standard sftp commands such as put (to the server) and get (to the client).
    b) For example, to transfer a file named myfile from your local directory to the z/OS home directory, you would use: **put myfile [z/OS_user_home_directory]/**
4. **Disconnect**: Upon finishing, exit the sftp session.

Architecture and Security Protocols

SHARE
EDUCATE • NETWORK • INFLUENCE

Transport Layer Protocol

- Secure low, level protocol
- Provides server authentication, confidentiality and integrity protection
- Runs over TCP/IP connection, may also be used on top of any other reliable data stream

Authentication Protocol

- Host based authentication at this protocol level
- Does not preform user authentication
- Higher level protocol can be designed on top for user authentication

Protocol Design

- Simple and flexible
- Allows parameter negotiation
- Minimizes the number of round trips

Negotiated Parameter

- Key exchange methods
- Public key algorithm
- Symmetric encryption algorithm
- Message authentication algorithm
- Hash algorithm

Client-Server Negotiation

Client

- SSH_FXP_INIT packet (client to server):
  - uint32 version
- Version 3 allowed extensions in SSH_FXP_INIT
  - Caused interoperability issues with v1/v2 servers
- This protocol version:
  - Use SSH_FXP_EXTENDED for extensions
    - After version exchange
  - Don't include extensions in version packet
    - Prevents issues with older servers

Server

- SSH_FXP_VERSION packet (server to client):
  - uint32 version
    - Protocol version supported by the server
    - Version number received from the client
  - <extension data>
    - May be empty
    - Or a sequence of string extension_name and string extension_data pair
      - Both strings must be present if one is
    - 'extension_data' string may be of zero length
- Extension data indicates extensions to the baseline protocol
  - 'extension_name' identifies the extension name
    - Format: "name@domain"
    - Domain is the DNS domain name of the organization defining the extension
    - Additional names may be defined by the IETF
  - Implementations must silently ignore unrecognized extensions

Configuring Open SSH Deamon

- The SSH daemon (sshd) must be run in the **POSIX C locale**
  - This usually happens automatically without user intervention
  - Alternate locale could be unintentionally selected through the shell profile of the user ID invoking the daemon or via the **ENVAR run-time option** in CEEPRMxx member of SYS1.PARMLIB

- To ensure the daemon runs in the POSIX C locale:
  - Use **STDENV** in the BPXBATCH job that starts the daemon
  - This explicitly sets `LC_ALL=C` to enforce the required locale

- Key points to remember:
  - Running sshd in the POSIX C locale is a security requirement
  - Accidentally using a different locale could introduce vulnerabilities
  - Properly configuring STDENV in the BPXBATCH job helps maintain compliance

Comparison

Comparison Chart

	FTP w/ no security RCF959	FTPS w/ SSL/TLS RCF959 + RCF4217	sftp
User ID and password protection	NO	YES	YES
Data protection (during file transfer)	NO	YES	YES
z/OS UNIX file support	YES	YES	YES
z/OS MVS dataset	YES	YES	NO (there are add-ons)
z System Hardware encryption capabilities	n/a	YES	YES (random number gen)
Partner authentication via locally stored copies of public keys	n/a	NO	YES
Partner authentication via X509 Certificates	n/a	YES	NO
Use SAF key rings and/or ICSF	n/a	YES	YES
FIPS 140-2 Mode	n/a	YES	NO
Mutual authentication supported	n/a	YES	YES

Resources

z/OS Open SSH Resources

- z/OS OpenSSH is part of z/OS base in Unix System Services
  - Starting in z/OS V2.2
  - z/OS OpenSSH User's Guide, SC27-6806

- OpenSSH was part of IBM Ported Tools for z/OS
  - Prior to z/OS V2.2
    - IBM Ported Tools for z/OS User's Guide (SA22-7985-06)
    - OpenSSH User's Guide (SA23-2246-02)

- The Internet Engineering Task Force (http://www.ietf.org/)
  - Four main SECSH internet drafts are:
    - SSH Transport Layer Protocol
      - draft-ietf-secsh-transport-17.txt
    - SSH Authentication Protocol
      - draft-ietf-secsh-userauth-20.txt
    - SSH Protocol Architecture
      - draft-ietf-secsh-architecture-15.5.txt
    - SSH File Transfer Protocol
      - draft-ietf-secsh-filexfer-05.txt

Your feedback is important!

SHARE
EDUCATE • NETWORK • INFLUENCE

Submit a session evaluation for each session you attend:

www.share.org/evaluation