IBM SWG – Enterprise Networking Solutions

GUIDE SHARE EUROPE
1959-2009
50 YEARS
OF GIVE AND TAKE
UK REGION

Safe and secure transfers with z/OS FTP

Alfred B Christensen – alfredch@us.ibm.com
Raleigh, NC, US

GSE UK Conference 2009

Safe and secure transfers with z/OS FTP

Date and time:	Thursday 5th November, 2009 from 09:00 to 10:00
Program:	Network Management working group
Speaker:	Alfred B Christensen, IBM
Abstract:	FTP is a readily available, convenient, and inexpensive technology to transfers files and data sets between z/OS and a virtually unlimited number of other operating system platforms. FTP is not a bad technology, as some recent press might lead you to believe. FTP can be misused and cause problems if the FTP service isn't properly set up to prevent potential security exposures. This session will explore a wide range of aspects related to how FTP works on z/OS. The session will reveal 'hidden gems' of FTP on z/OS and will look at a set of usage scenarios, providing suggestions on how to best exploit selected features of the z/OS FTP technology. The session will especially focus on how you can secure both the FTP environment itself and the individual data transfers that z/OS FTP participates in both as a client and as a server.

One-day IBM ITSO workshop on how to assess, plan for, and implement the z/OS V1R11 Communications Server enhancements:

System z Networking Technologies Update, WRZ005GB
Starts 10th November 2009 for 1 day in Bedfont Lakes, U.K.
Contact Name: Khaled Ibrahim - Khaled_Ibrahim@uk.ibm.com

http://www.redbooks.ibm.com/projects.nsf/WorkshopIndex/

Trademarks, notices, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries or both:

- Advanced Peer-to-Peer Networking®
- AIX®
- alphaWorks®
- AnyNet®
- AS/400®
- BladeCenter®
- Candle®
- CICS®
- DB2 Connect
- DB2®
- DRDA®
- e-business on demand®
- e-business (logo)
- e business(logo)®
- ESCON®
- FICON®
- GDDM®
- HiperSockets
- HPR Channel Connectivity
- HyperSwap
- i5/OS (logo)
- i5/OS®
- IBM (logo)®
- IBM®
- IMS
- IP PrintWay
- IPDS
- iSeries
- LANDP®
- Language Environment®
- MQSeries®
- MVS
- NetView®
- OMEGAMON®
- Open Power
- OpenPower
- Operating System/2®
- Operating System/400®
- OS/2®
- OS/390®
- OS/400®
- Parallel Sysplex®
- PR/SM
- pSeries®
- RACF®
- Rational Suite®
- Rational®
- Redbooks
- Redbooks (logo)
- Sysplex Timer®
- System i5
- System p5
- System x
- System z
- System z9
- Tivoli (logo)®
- Tivoli®
- VTAM®
- WebSphere®
- xSeries®
- z9
- zSeries®
- z/Architecture
- z/OS®
- z/VM®
- z/VSE

- Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
- Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
- Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Linux is a trademark of Linus Torvalds in the United States, other countries, or both.
- Red Hat is a trademark of Red Hat, Inc.
- SUSE® LINUX Professional 9.2 from Novell®
- Other company, product, or service names may be trademarks or service marks of others.
- This information is for planning purposes only. The information herein is subject to change before the products described become generally available.
- Disclaimer: All statements regarding IBM future direction or intent, including current product plans, are subject to change or withdrawal without notice and represent goals and objectives only. All information is provided for informational purposes only, on an "as is" basis, without warranty of any kind.

All performance data contained in this publication was obtained in the specific operating environment and under the conditions described and is presented as an illustration. Performance obtained in other operating environments may vary and customers should conduct their own testing.

Refer to www.ibm.com/legal/us for further legal information.

Agenda

- ❑ FTP and Security – an oxymoron?
- ❑ z/OS FTP – local security
- ❑ Secure FTP: network traversal challenges and solutions
- ❑ Secure FTP: Keys and certificates overview
- ❑ Appendix:
  - ❑ RACDCERT commands to create keys and certificates for a secure z/OS FTP server
  - ❑ Secure z/OS FTP server FTP.DATA and associated ATTLS policy

Disclaimer: All statements regarding IBM future direction or intent, including current product plans, are subject to change or withdrawal without notice and represent goals and objectives only. All information is provided for informational purposes only, on an "as is" basis, without warranty of any kind.

Safe and Secure Transfers with z/OS FTP

FTP and Security – an oxymoron?

Let's try and clear a little common confusion from the start

- **FTP:**
  - Also referred to as RFC959 FTP or “normal” FTP
  - The FTP protocol we all know and have used for years.
  - The FTP protocol has been extended numerous times since the original RFC 959 was issued in 1985
    - Specific support for both Kerberos-based and SSL/TLS-based security has been added to the FTP protocol
      - RFC4217 "Securing FTP with TLS"
  - What the z/OS CS FTP client and server have supported through many years
    - An RFC959 FTP client talks to an RFC959 FTP server, and not to an SFTP server

RFC959 FTP

- **SFTP:**
  - Secure Shell file transfer protocol
    - A sub-protocol of SSH (Secure Shell)
    - Supported on z/OS by "IBM Ported tools for z/OS" and at least two ISV products
    - Has nothing to do with RFC959 FTP - incompatible protocols
    - An SFTP client talks to an SFTP server and not an RFC959 FTP server

Secure Shell FTP

- **FTPS:**
  - Also referred to as RFC4217 FTP, FTP AUTH-TLS, or FTP AUTH-SSL
  - Secure RFC959 FTP using a standard security mechanism, such as Kerberos or SSL/TLS
    - RFC4217 "Securing FTP with TLS"
  - The normal FTP protocol but extended with full network security (authentication, data integrity, and data privacy)
  - Both control connection and data connection can be secured
    - No user IDs or password flowing in the clear

RFC4217 FTP

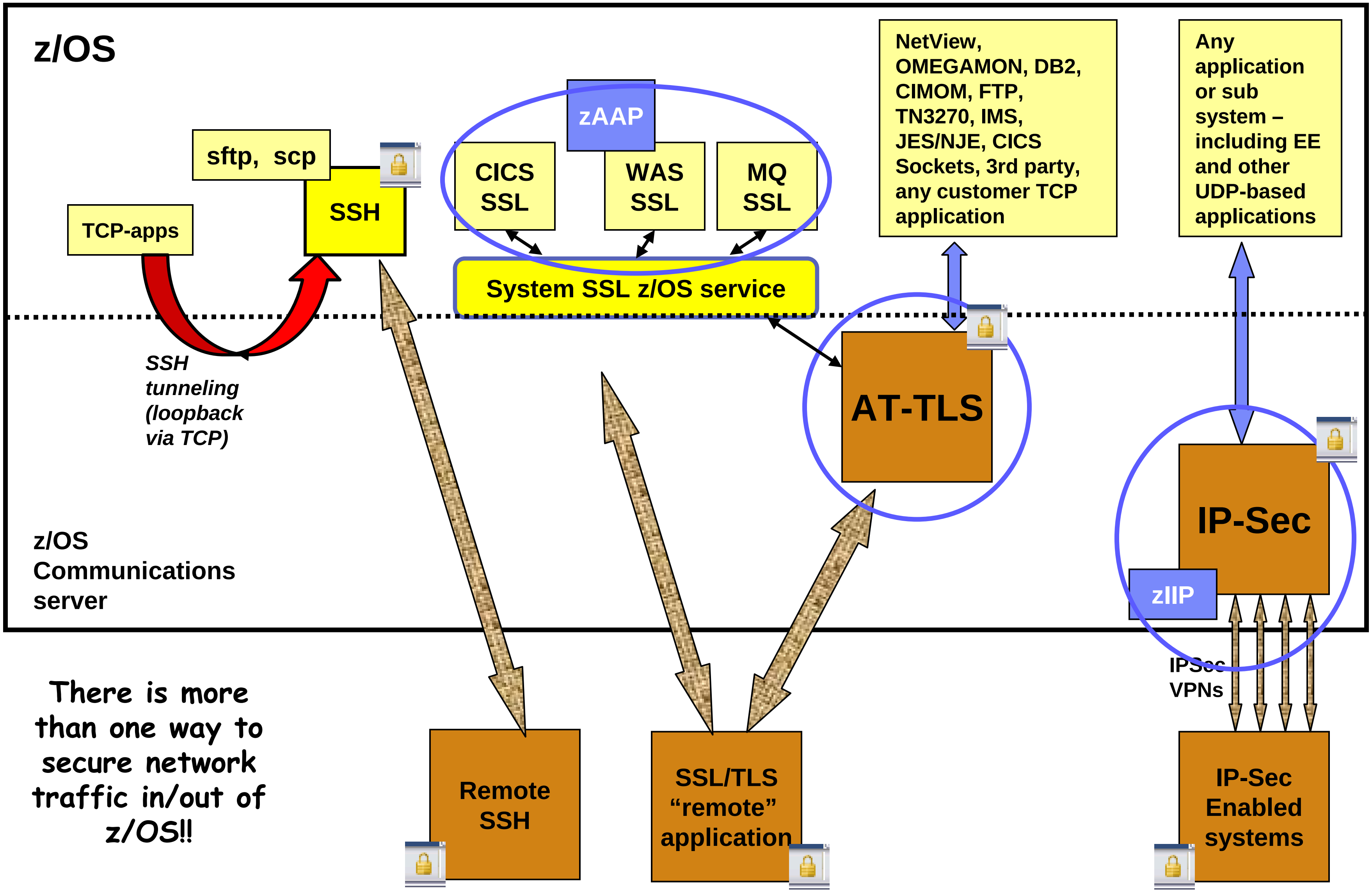
z/OS network encryption introduction

- **General types of network encryption:**
  - IPSec VPNs (Virtual Private Networks) – system to system, fully transparent to applications
  - SSL (Secure Sockets Layer) – application to application (TCP only)
    - The IETF standardized SSLv3 under the name TLS (Transport Layer Security).
    - SSL/TLS services are provided by the System SSL z/OS component
  - SSH – Secure Shell can to some degree be considered general (TCP only)
    - It supports SSH-specific applications (sftp, scp, SSH login)
    - It also support general TCP applications through tunnelling over a local connection

- **Two ways SSL/TLS has been implemented on z/OS:**
  - Application or subsystem layer encryption (per connection)
  - Network layer encryption (also per connection), but using “common service” transparent to the z/OS application or subsystem in z/OS V1R7+

- **IPSec on z/OS:**
  - “System to system” encryption, transparent to all applications and subsystems (including UDP traffic, which Enterprise Extender uses)
  - IPSec can use zIIP today (z/OS V1R8+)
  - Use of zIIP depends on network traffic – the more traffic, the higher the zIIP usage

z/OS network encryption technology overview


z/OS
sftp, scp
TCP-apps
SSH
SSH tunneling (loopback via TCP)
zAAP
CICS SSL
WAS SSL
MQ SSL
System SSL z/OS service
NetView, OMEGAMON, DB2, CIMOM, FTP, TN3270, IMS, JES/NJE, CICS Sockets, 3rd party, any customer TCP application
Any application or sub system – including EE and other UDP-based applications
AT-TLS
IP-Sec
zIIP
z/OS Communications server
IPSec VPNs
Remote SSH
SSL/TLS “remote” application
IP-Sec Enabled systems

There is more than one way to secure network traffic in/out of z/OS!!

A quick comparison of selected z/OS file transfer technologies from a security perspective

	FTP With no security RFC959	FTPS FTP w. SSL/TLS RFC959 + RFC4217	FTP FTP w. IPSec Any RFC level	SFTP As implemented by IBM Ported Tools
User ID and password protection	No	Yes	Yes	Yes
Data protection (the file being transferred)	No	Yes	Yes	Yes
z/OS UNIX file support	Yes	Yes	Yes	Yes
z/OS MVS data set support	Yes	Yes	Yes	No
Use of System z hardware encryption technologies	n/a	Yes	Yes	No
Partner authentication via locally stored copies of public keys	n/a	No	Yes (pre-shared key)	Yes
Partner authentication via X509 certificates	n/a	Yes	Yes	No
Use of SAF key rings and/or ICSF	n/a	Yes	Yes	No
FIPS 140-2 mode	n/a	Yes (z/OS V1R11)	No	No
Mutual authentication supported	n/a	Yes	Yes (at an IP address level)	Yes

So what are some of the arguments against using FTP for secure file transfers? (Part 1 of 2)

- **"FTP is not secure"**
  - RFC959 FTP is not secure, but RFC4217 FTP is as secure as any other secure file transfer technology
    - Secures both the control connection (user ID and password) and the data connection (the file being transferred)
- **"FTP lacks automation capabilities and has only a manual user interface"**
  - Almost all operating system platforms support some form of a client FTP programming interface including z/OS
  - The z/OS FTP server supports multiple exit points, SMF records, NMI interface events, and activity logging to syslogd
    - Enabling management solutions to be added on top of FTP
- **"A difficult protocol to punch holes in firewalls for – especially when using RFC4217 FTP or FTP with IPSec"**
  - This is correct
  - This is a major issue with use of FTP for inter-company file transfers
  - Addressed by some recent extensions to the FTP protocol

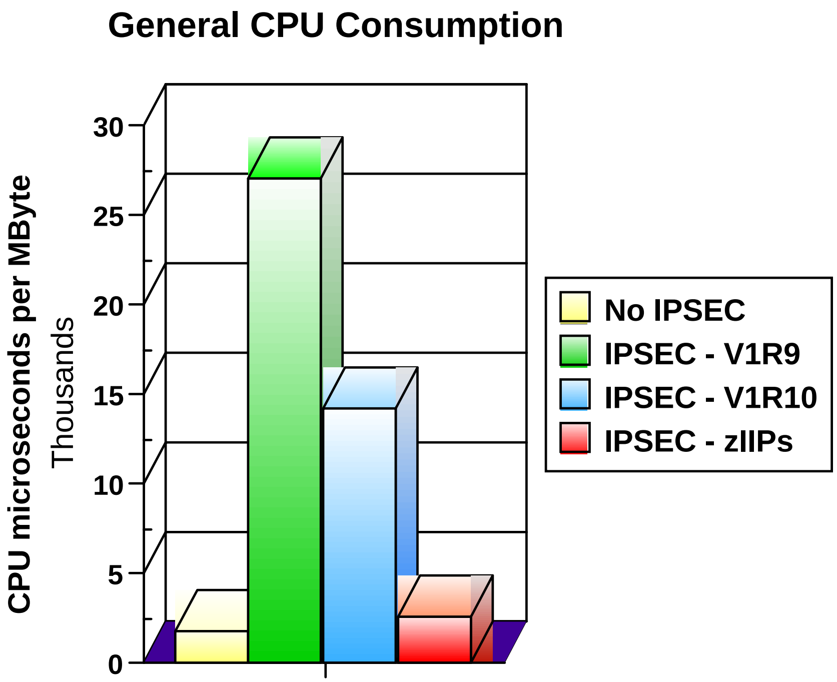
So what are some of the arguments against using FTP for secure file transfers? (Part 2 of 2)

- **“FTP doesn’t provide a management function”**
  - That is correct.
  - FTP provides the raw file transfer function, not a file transfer scheduling, execution, monitoring, and auditing capability
    - FTP does provide management interfaces for management functions
  - Separate offerings from many vendors provide such management functions that may or may not use FTP as the underlying file transfer technology
    - FTP in itself is not a “managed file transfer” offering

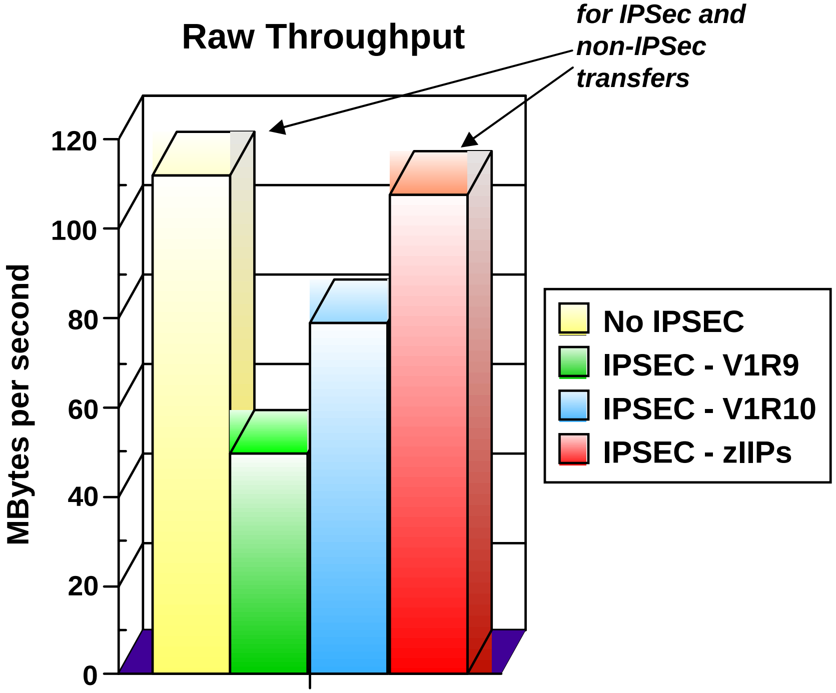
- **“FTP isn’t cost competitive”**
  - That depends.
  - First of all, FTP is a “free” component on almost all operating system platforms
    - It is readily available for no extra software costs
  - Second, FTP is a very low overhead protocol that on z/OS benefits from a long range of bulk transfer performance functions
  - RFC4217 FTP and FTP with IPSec use CPACF and other hardware encryption accelerators on System z
  - When securing FTP with IPSec, zIIP processors are used to offload much of the CPU overhead associated with security

zIIP-assisted IPSec - outbound bulk transfer workload performance

- Example:
  - 10 concurrent streaming outbound sessions using AES encryption and SHA authentication
- Same overall picture for inbound streaming workload



General CPU Consumption

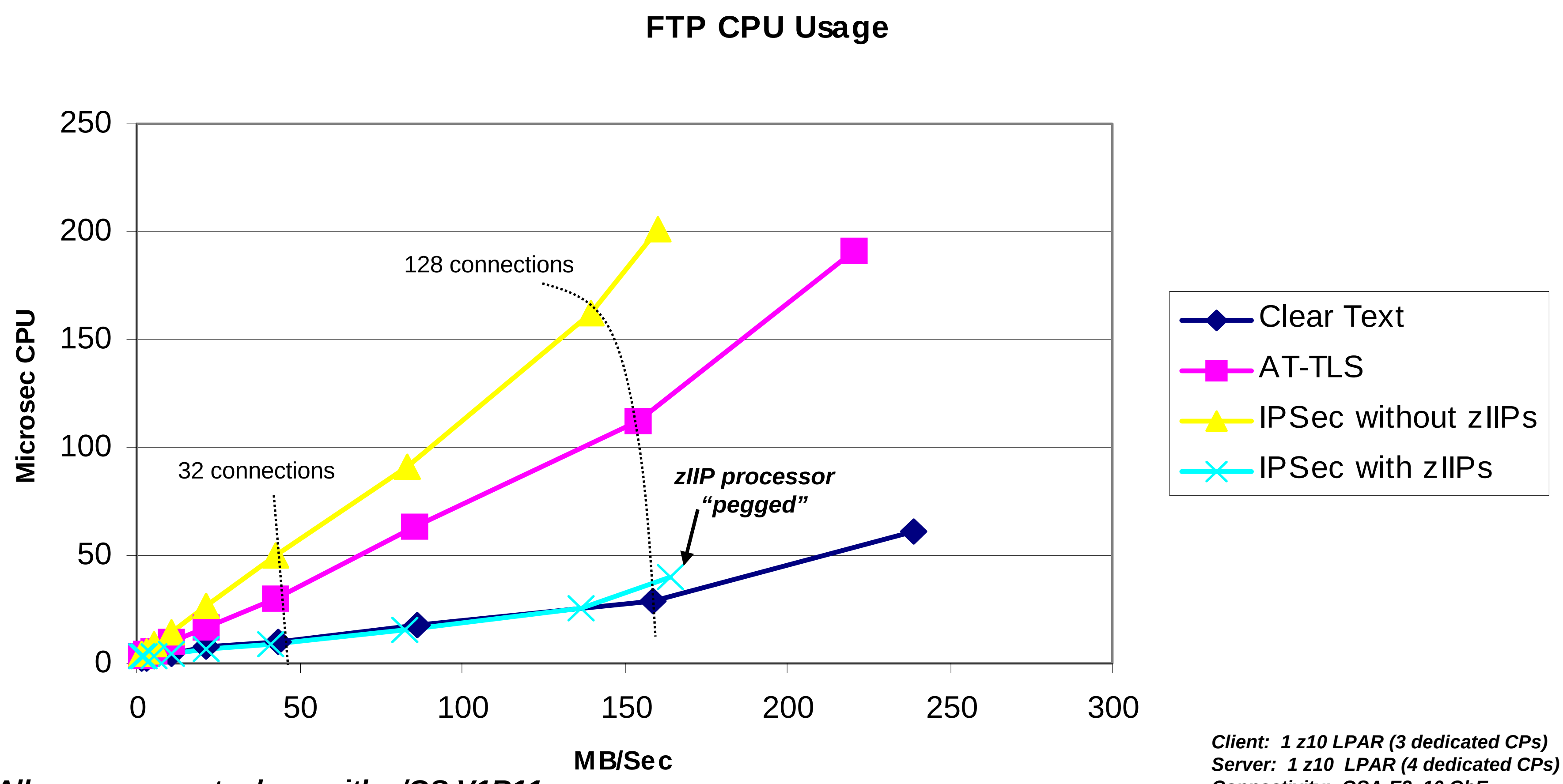


Raw Throughput

Very close to equal throughput for IPSec and non-IPSec transfers

All performance data contained in this publication was obtained in the specific operating environment and under the conditions described and is presented as an illustration. Performance obtained in other operating environments may vary and customers should conduct their own testing.

Comparing FTP Server CPU usage with and without security



All measurements done with z/OS V1R11
Outbound Data (Gets) to an MVS client
3DES encryption with SHA authentication
From 1 to 128 parallel connections
Highest throughput numbers obtained with 0 think-time

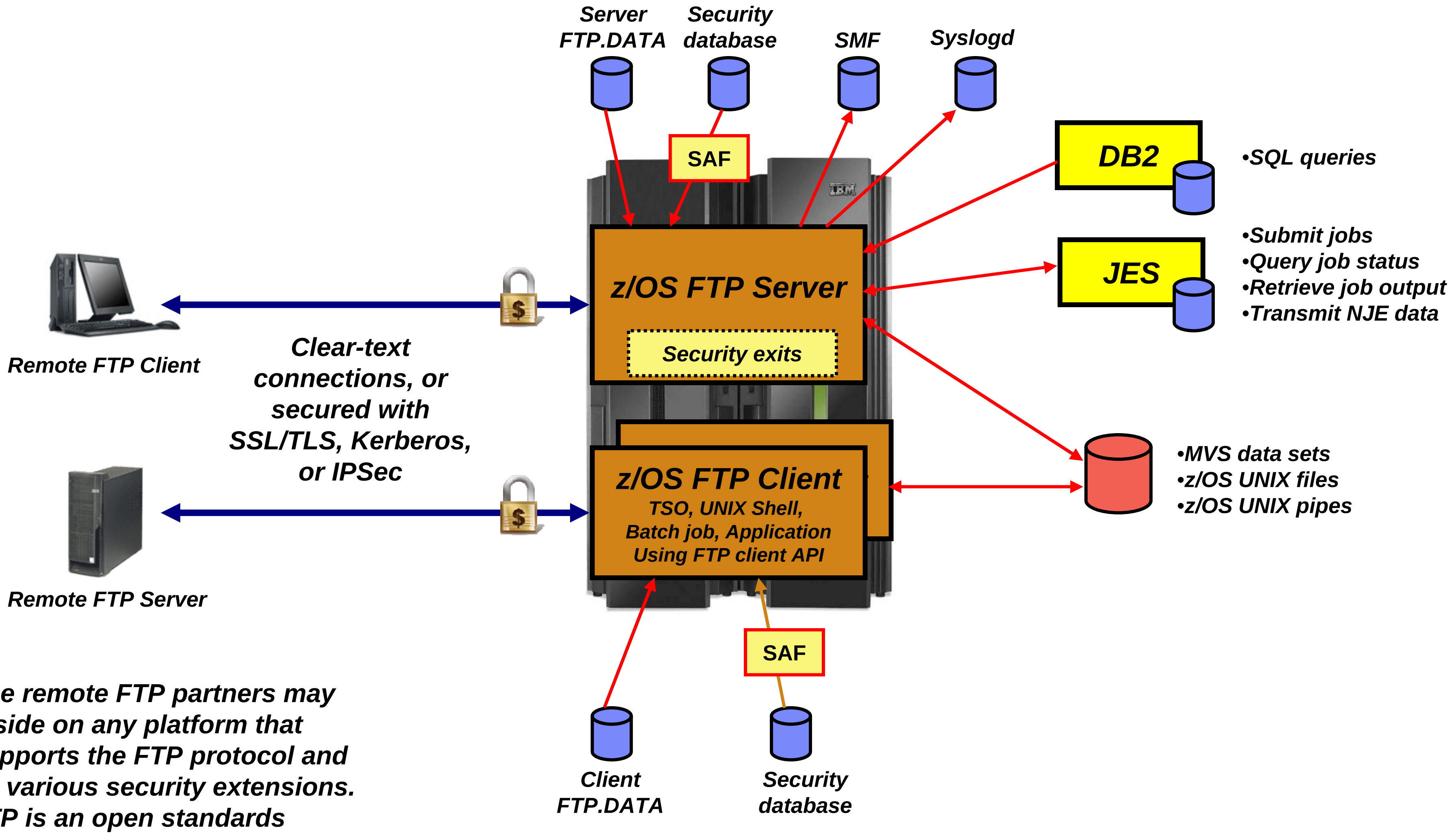
Client: 1 z10 LPAR (3 dedicated CPs)
Server: 1 z10 LPAR (4 dedicated CPs)
Connectivity: OSA-E3 10 GbE
Encryption/Authentication: 3DES/SHA
Transaction: 1 byte / 2 MB
Target data sets: MVS data sets on 3390 DASD
Think time: 1500 ms
Number of connections: 1 to 128
Driver tool: AWM

All performance data contained in this publication was obtained in the specific operating environment and under the conditions described and is presented as an illustration. Performance obtained in other operating environments may vary and customers should conduct their own testing.

Safe and Secure Transfers with z/OS FTP

z/OS FTP – local security

z/OS FTP – the big picture



The remote FTP partners may reside on any platform that supports the FTP protocol and its various security extensions. FTP is an open standards protocol.

Securing the local z/OS FTP server

- **Basic platform security setup is a pre-requisite**
  - Users defined with proper MVS data set access protection
  - z/OS UNIX files defined with proper owning user and group along with user/group/world access permissions
  - Etc.
- **FTP server-specific SAF resource definitions**
  - Via SERVAUTH resource profiles
- **Security-related options in the server's FTP.DATA**
  - Controlling various aspects of how the FTP server reacts to selected requests, such as a request for anonymous access
- **Optional security exits in the FTP server**
  - Can be implemented to provide vary granular levels of controls in the FTP server

Selected SAF resource definitions in the SERVAUTH class

- **EZB.PORTACCESS.*sysname.tcpname.port_safname***
  - Controls ability for a started task user ID to establish itself as a server on the matching port number in the TCP/IP Profile port reservation section

- **EZB.FTP.*sysname.ftpdaemonname*.PORT*xxxxx***
  - Controls ability to log into an FTP server (control port number) based on the SAF user ID that is being used to log in
  - Initially used for SSL/TLS connections if SECURE_LOGIN VERIFY_USER was coded in the FTP server's FTP.DATA
  - Can be enforced for all types of connections by coding VERIFYUSER TRUE in the server's FTP.DATA - (This support was added in z/OS V1R10)

- **EZB.FTP.*sysname.ftpdname*.SITE.DUMP** and **EZB.FTP.*sysname.ftpdname*.SITE.DEBUG**
  - Provides ability to restrict usage of SITE DUMP and DEBUG commands (commands may generate large amount of output)

- **EZB.FTP.*sysname.ftpdaemonname*.ACCESS.HFS**
  - Provides ability to generally restrict FTP user access to the z/OS UNIX file system

Selected security options in the FTP server's FTP.DATA

- **ANONYMOUS**
  - Controls the ability to log into your FTP server as an anonymous user
  - If the ANONYMOUS option is not included in the server's FTP.DATA, anonymous access is disabled
  - Disabled by default – keep it that way, unless you have specific need for it.
    - If you do enable ANONYMOUS, make sure to change the default value of 1 on the ANONYMOUSLEVEL option to 3
    - Also, verify the settings of all the options that start with ANONYMOUS.. – there are a total of 8 including the ANONYMOUS option itself
    - Use the supplied shell script to build a specific z/OS UNIX file system directory structure for anonymous access
    - EMAILADDRCHECK is a syntax check only of the entered email address
- **DEBUGONSITE and DUMPONSITE**
  - Controls the ability to enable dump and debug SITE command options
  - If you set these to TRUE, make sure you define the corresponding SERVAUTH profiles so only authorized users can issue these two SITE command options
- **PORTCOMMAND, PORTCOMMANDPORT, PORTCOMMANDIPADDR, and PASSIVEDATACONN**
  - Control the ability of your FTP server to participate in three-way proxy mode.
  - See next page for more details

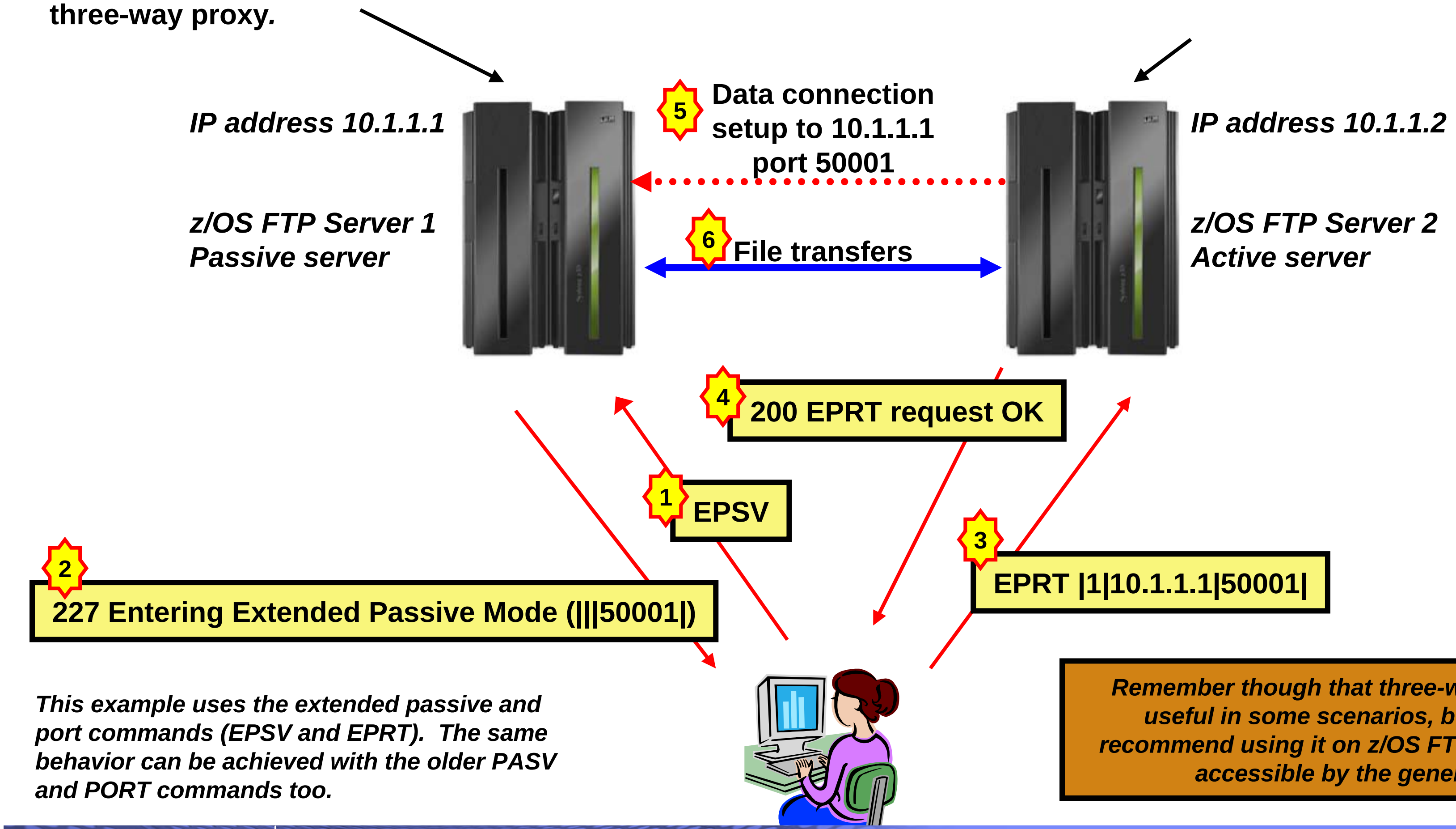
How to disable three-way proxy FTP operations

PASSIVEDATACONN says what this server is to do when it receives a data connection setup request from a source IP address that isn't the same as the FTP client IP address.

Set PASSIVEDATACONN to NOREDIRECT to disable three-way proxy*.*

PORTCOMMANDIPADDR says what this server is to do when it receives a PORT or EPRT command with an IP address that isn't the same as the FTP client IP address.

Set PORTCOMMANDIPADDR to NOREDIRECT to disable three-way proxy.



IP address 10.1.1.1

z/OS FTP Server 1
Passive server

IP address 10.1.1.2

z/OS FTP Server 2
Active server

5 Data connection setup to 10.1.1.1 port 50001

6 File transfers

1 EPSV

2 227 Entering Extended Passive Mode (|||50001|)

3 EPRT |1|10.1.1.1|50001|

4 200 EPRT request OK

This example uses the extended passive and port commands (EPSV and EPRT). The same behavior can be achieved with the older PASV and PORT commands too.

Remember though that three-way proxy can be useful in some scenarios, but I would not recommend using it on z/OS FTP servers that are accessible by the general users.

Selected security options in the FTP server's FTP.DATA - continued

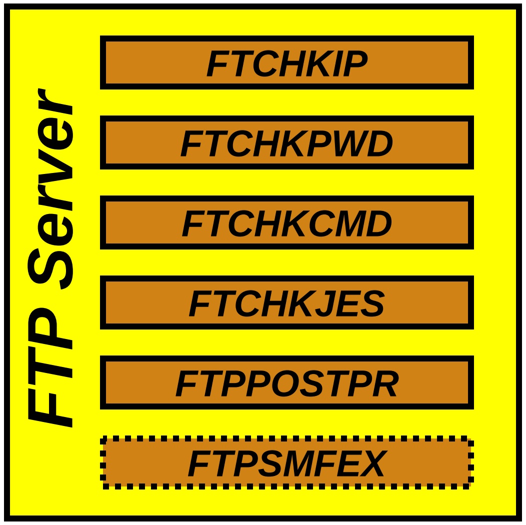
- **REPLYSECURITYLEVEL**
  - Controls how much identification information is sent on the initial 220 greeting message from the FTP server, and also how much detail is returned when MVS data set contention occurs.
  - Default is no restrictions (level 0).
  - If your auditors request you to send as little information as possible, use a setting of 1 on this option
    - Level 0: 220-FTPABC1 IBM FTP CS V1R11 at MVS098, 16:42:51 on 2009-05-24.
    - Level 1: 220-IBM FTP, 16:45:57 on 2009-05-24.
- **ACCESSERRMSG**
  - To prevent details of failed log in attempts to be returned to the FTP client user, set this option to FALSE (which is the default).
  - You may change it to TRUE in an internal-only shop if you want your users to receive details about their failed log in attempt.
- **SECURE_...**
  - There are a number of options that start with SECURE_ - they are all used to control the ability of the FTP server to accept secure connections (SSL/TLS or Kerberos)

Selected security options in the FTP server's FTP.DATA - continued

- **VERIFYUSER**
  - Discussed earlier – extends SAF check of all users' ability to access the server's control port number
    - EZB.FTP.sysname.ftpdaemonname.PORTxxxxx
- **PASSIVEDATAPORTS**
  - Controls which range of port numbers the server may use for passive mode data connections

If it is a few years ago you created your server's FTP.DATA data set, I recommend recreating it based on the FTPSDATA member in hlq.SEZAINST – many new options have been added over the last releases and all are included in this sample member for documentation purposes.

FTP server security exit points – extending FTP server security



FTP Server

FTCHKIP	*Accept/reject connections based on client and server IP address and port information*
FTCHKPWD	*Accept/reject login based on client user ID and/or password*
FTCHKCMD	*Accept/reject/modify individual FTP commands and their arguments*
FTCHKJES	*Accept/reject submission of a job based on analyzing records of job to be submitted*
FTPPOSTPR	*Initiate file transfer post processing based on result of file transfer*
FTPSMFEX	*Accept/reject writing of old SMF118 records (no longer recommended)*

- If these exits routines are present they will be loaded and called at the defined exit points
- The FTCHKIP exit is called by the FTP daemon, while the others are called by the FTP server (after the new address space has been created)
- The command check routine is the most widely used. It has information about the current command from the client, what the current working directory is, what file-type we are using, etc. It may reject the command or it may modify the command options, such as the file or data set name on a STOR or RETR command. If it does reject the command, it can also return the text that will be returned to the client in the 500 reply
- The FTCHKCMD exit executes under the logged in user's user ID. Installation-defined SAF resource definitions can be checked in that routine if needed
- The exits are normally coded in assembler, but we have seen examples where they were coded in C.

FTP server security exit details

Exit point	Called by	Called when	Main input	Possible actions
FTCHKIP	Daemon address space	When control connection is being accepted by the FTP daemon	Client and server IP addresses and ports	Accept or reject connection setup
FTCHKPWD	Server address space	When the client user sends the PASS command	IP addresses and ports, client user ID and password	Accept or reject login request
FTCHKCMD	Server address space	For every command received over the control connection	IP addresses and ports, client user ID, directory type, file type, current directory, and the FTP command and arguments	Accept, reject, or modify the FTP command
FTCHKJES	Server address space	For every record in a job that is being submitted to JES	IP addresses and ports, the full JES input record	Accept or reject the job submission
FTPOSTPR	Server address space	For every completed file transfer operation	IP addresses and ports, plus details about the completed file transfer	Initiate post processing

Samples for all in hlq.SEZAINST

Securing the local z/OS FTP client

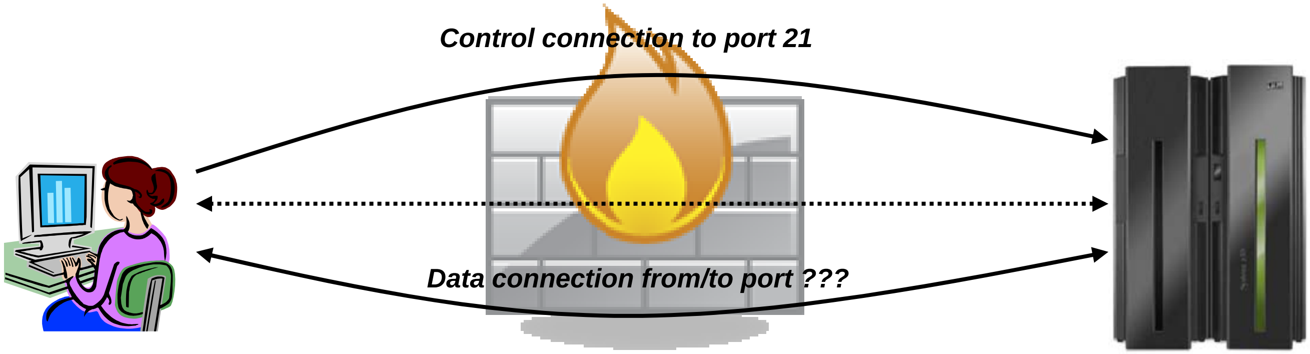
- **Basic platform security setup is a pre-requisite**
  - Users defined with proper MVS data set access protection
  - z/OS UNIX files defined with proper user/group/world access permissions
  - Etc.
- **FTP server-specific SAF resource definitions**
  - None for the FTP client
- **Security-related options in the client's FTP.DATA**
  - Not really any
- **Optional security exits**
  - No exit points in the z/OS FTP client (but requirement to have one has been dutifully noted)

There really isn't much you can do in this area short of protecting the FTP client program itself.

Safe and Secure Transfers with z/OS FTP

Secure FTP: network traversal challenges and solutions

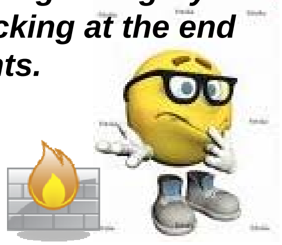
Firewalls and FTP


Control connection to port 21
Data connection from/to port ???

- *Port-based filter rules*
- *Network Address Translation (NAT)*

- **Port-based filter rules – in particular dynamic port rules**
  - FTP control connection is no problem - pre-defined server port number (default 21)
  - Data connection port number (or direction) is not pre-defined, but dynamically negotiated between the FTP client and server
    - The firewall does "deep inspection" (peeks into) the FTP control connection to learn about the negotiated ports and the direction for the data connection
- **NAT**
  - FTP control connection is no problem – only IP headers need translation
  - PORT command and PASV reply refers to local (intranet) IP addresses
    - Firewall needs to do "deep inspection" of the FTP control connection to locate and modify the IP address information in the PORT command and the PASV reply

Deep inspection and data modification is impossible when the data on the FTP control connection is secured through encryption and message integrity checking at the end points.



So what if I need both FTP security and firewalls?

I am a firewall who wants to inspect the FTP control connection data !

No encryption:

SrcIP	DestIP	SrcPort	DestPort	Data
192.168.100.1	192.168.1.1	21	50001	227 Entering Passive Mode (192.168.100.1, 50010)

SSL/TLS encryption:

SrcIP	DestIP	SrcPort	DestPort	Data
192.168.100.1	192.168.1.1	21	50001	@%$#*&&^^!:"J)*GVM><

IPSec encryption:

SrcIP	DestIP	SrcPort	DestPort	Data
192.168.100.1	192.168.1.1	>::"	*&hU$$$$	@%$#dd*&&^s^!:"J)*bGVM>(*hhgvvv<

IP header encryption varies based on transport/tunnel mode, and AH/ESP protocol

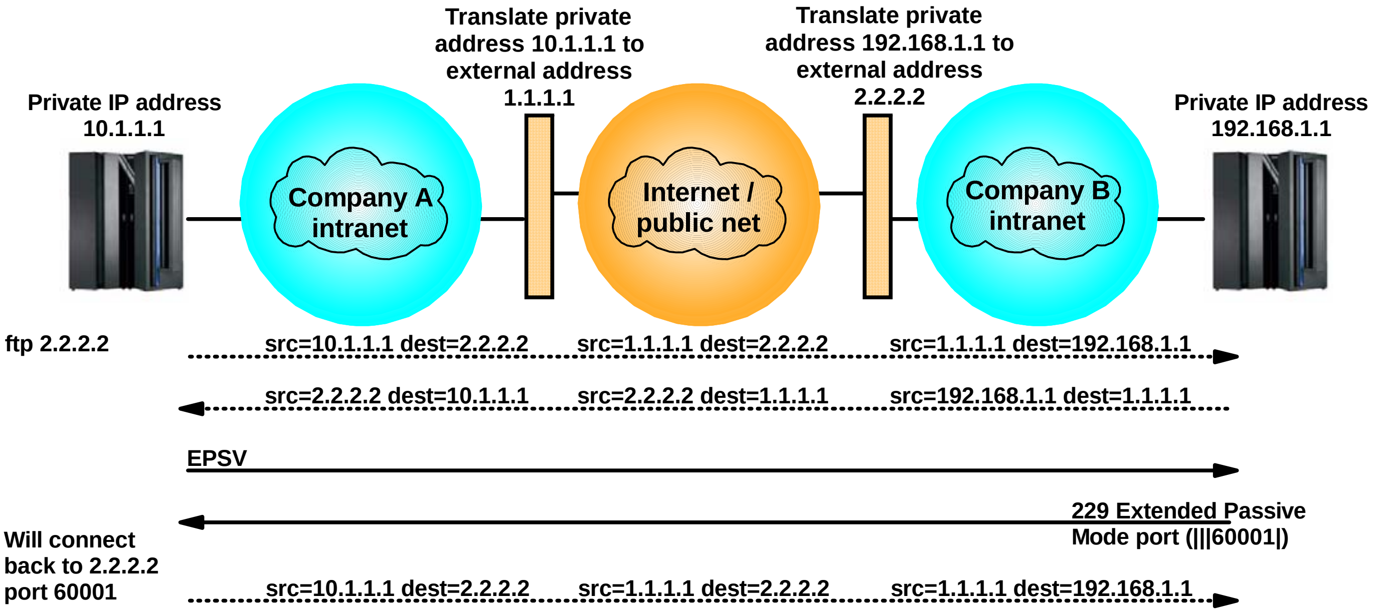
- **No firewalls – no problems**
  - Dream on …
- **No FTP security, but firewalls**
  - Firewalls manage port filtering by deep inspection
  - Firewalls manage NAT by deep inspection and modification of data on the control connection
- **FTP security, and firewalls**
  - Requires a bit of ingenuity !!!!
  - See the following pages.

RFC 2428: FTP Extensions for IPv6 and NATs

- **Extended passive mode (EPSV) will solve NAT problems for secure FTP sessions**
  – If using z/OS FTP client to a server that does not support EPSV, code PASSIVEIGNOREADDR TRUE in the FTP client's FTP.DATA

- **The EPSV reply does not include an IP address, but only a port number**
  – The FTP client will connect to the same IP address it used for the control connection

- **The EPSV and the accompanying extended port command (EPRT) are also used to enable IPv6 support in FTP**
  – Used with IPv4, the EPSV command provides NAT firewall relief



Translate private address 10.1.1.1 to external address 1.1.1.1

Translate private address 192.168.1.1 to external address 2.2.2.2

Private IP address 10.1.1.1

Company A intranet

Internet / public net

Company B intranet

Private IP address 192.168.1.1

ftp 2.2.2.2

src=10.1.1.1 dest=2.2.2.2 src=1.1.1.1 dest=2.2.2.2 src=1.1.1.1 dest=192.168.1.1

src=2.2.2.2 dest=10.1.1.1 src=2.2.2.2 dest=1.1.1.1 src=192.168.1.1 dest=1.1.1.1

EPSV

229 Extended Passive Mode port (|||60001|)

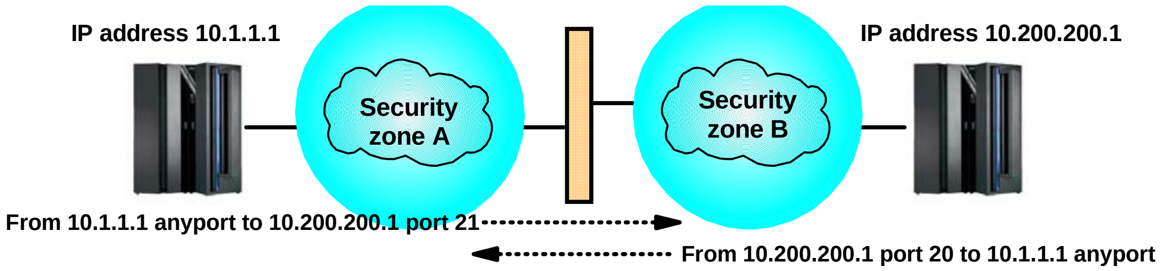
Will connect back to 2.2.2.2 port 60001

src=10.1.1.1 dest=2.2.2.2 src=1.1.1.1 dest=2.2.2.2 src=1.1.1.1 dest=192.168.1.1

RFC 2428 does not help with dynamic port-based filter rules in firewalls!

How to deal with static port-based filters in firewalls

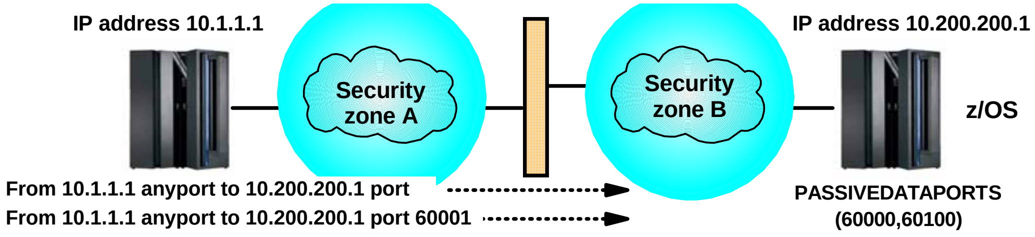
- **If you are able to use active mode FTP, the firewall filters can sometimes be managed:**
  - The control connection is permitted inbound to port 21
  - The data connection is permitted outbound from port 20
  - Will work for both standard active mode (PORT) and extended active mode (EPRT)


IP address 10.1.1.1
Security zone A
Security zone B
IP address 10.200.200.1
From 10.1.1.1 anyport to 10.200.200.1 port 21
From 10.200.200.1 port 20 to 10.1.1.1 anyport

Static firewall filters

- Connection setup from 10.1.1.1 any port to port 21 on 10.200.200.1 - permit
- Connection setup from 10.200.200.1 port 20 to 10.1.1.1 any port - permit

- **If you use passive mode FTP, and your server is a z/OS FTP server, you can predefine a range of port numbers to be used for passive mode data connections**
  - The control connection is permitted inbound to port 21
  - The data connection is permitted inbound to a port in a pre-defined range
  - Will work for both standard passive mode (PASV) and extended passive mode (EPSV)

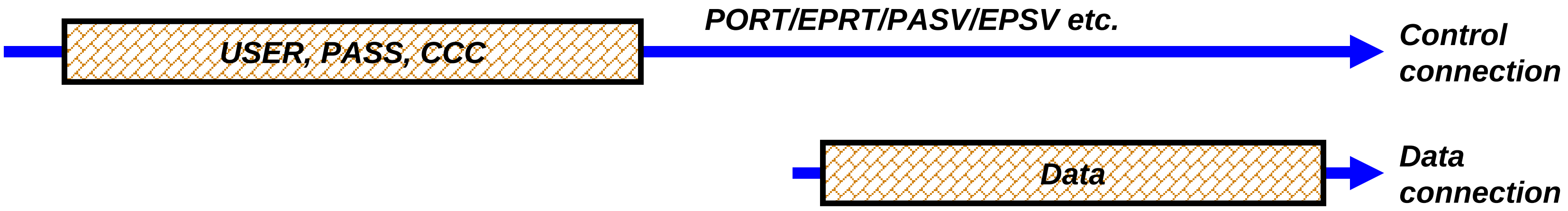


Static firewall filters

- Connection setup from 10.1.1.1 any port to port 21 on 10.200.200.1 - permit
- Connection setup from 10.1.1.1 any port to a port in the range from 60000 to 60100 on 10.200.200.1 - permit

How to deal with dynamic port-based filters in firewalls

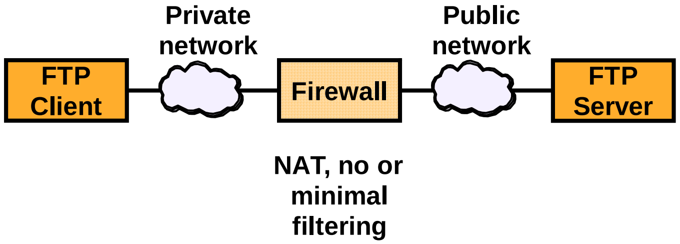
- **When using dynamic filters, the firewall enables (permits) ports based on IP address and/or port number information in the PORT/EPRT command or the PASV/EPSV reply**
  - The original FTP SSL/TLS draft RFC stated that the FTP control connection always had to be encrypted!
  - The final RFC (RFC 4217 "Securing FTP with TLS") relaxes on this requirement and implements a new Clear Command Channel (CCC) FTP command



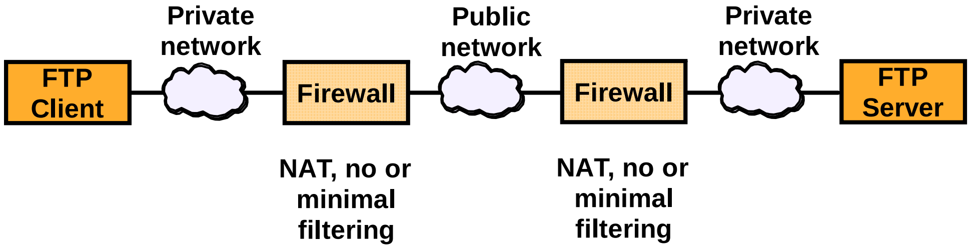
- **Both the FTP client and server need to support the CCC command according to RFC 4217**
  - Not all FTP clients and servers that support FTP SSL/TLS support the CCC command
    - z/OS added full support for the CCC command in z/OS V1R9 (both z/OS FTP client and server)
      - APAR PK26746 supplied this function for the z/OS FTP client in fall 2006 (back to z/OS V1R4)
  - For those products that claim support, some interoperability issues have been observed !
    - If you have problems getting CCC to work, try to specify TLSRFCLEVEL CCCNONOTIFY instead of TLSRFCLEVEL RFC4217 (applies to both z/OS FTP server and client)
      - CCCNONOTIFY supports a pre-RFC4217 level of the CCC command processing, which some FTP implementations are based upon
  - z/OS FTP server must have SECURE_CTRLCONN CLEAR configured to accept a CCC command

- **In general, the CCC command is a solution that solves SSL/TLS-enabled FTP issues with both NAT firewalls and filtering firewalls**

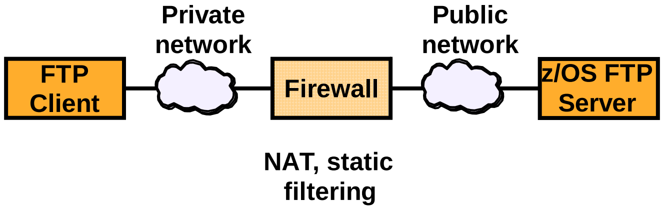
FTP and firewall topologies – part 1 of 2



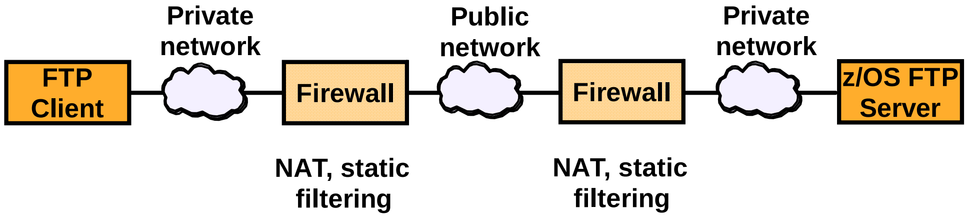
- ✓ *Normal passive mode (PASV) will usually work in such a topology.*
- ✓ *Extended passive mode (EPSV) will also work, but is not generally required.*



- ✓ *If your partner secure FTP product supports extended passive mode - use extended passive mode (EPSV) from the FTP client.*
- ✓ *If the FTP client is on z/OS (V1R11) and the partner secure FTP server product does not support EPSV, configure the PASSIVEIGNOREADDR option at your z/OS FTP client to simulate EPSV processing.*

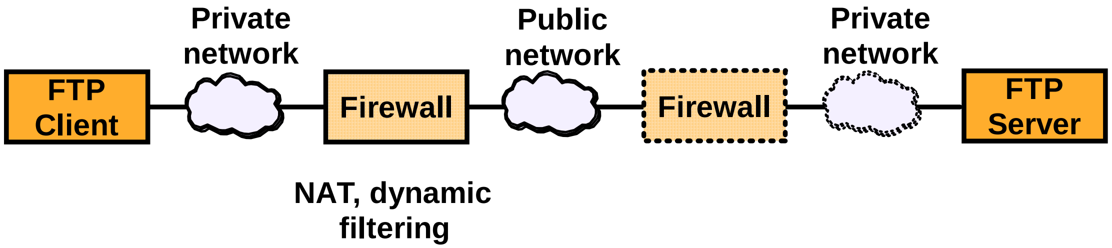


- ✓ *Use the PASSIVEDATAPORTS option on the z/OS FTP server to predefine a range of port numbers the z/OS FTP server may use for data connections.*
  - ✓ *Other FTP servers may have similar configuration capabilities*
- ✓ *Have your firewall administrator add static filter rules for the passive data port range.*
- ✓ *Normal passive mode (PASV) will usually work in such a topology, but extended passive mode (EPSV) can also be used if supported by the FTP client.*

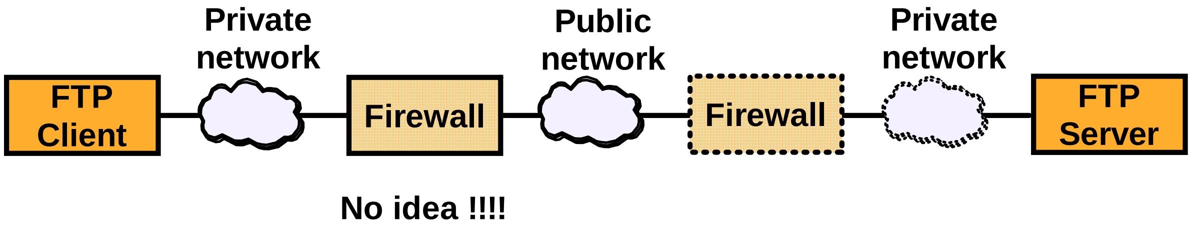


- ✓ *Use the PASSIVEDATAPORTS option on the z/OS FTP server to predefine a range of port numbers the z/OS FTP server may use for data connections.*
  - ✓ *Other FTP servers may have similar configuration capabilities*
- ✓ *Have your firewall administrator add static filter rules for the passive data port range.*
- ✓ *In this case, you must use extended passive mode.*
- ✓ *If the FTP client does not support extended passive mode, you will likely not get this scenario to work.*
- ✓ *If the FTP client is on z/OS (V1R11) and the partner secure FTP server product does not support EPSV, configure the PASSIVEIGNOREADDR option at your z/OS FTP client to simulate EPSV processing.*

FTP and firewall topologies – part 2 of 2



- ✓ *Use the CCC command from the FTP client.*
- ✓ *You will most likely not get this scenario to work without the CCC command support.*



- ✓ *Use the CCC command from the FTP client.*
- ✓ *You will most likely not get this scenario to work without the CCC command support*

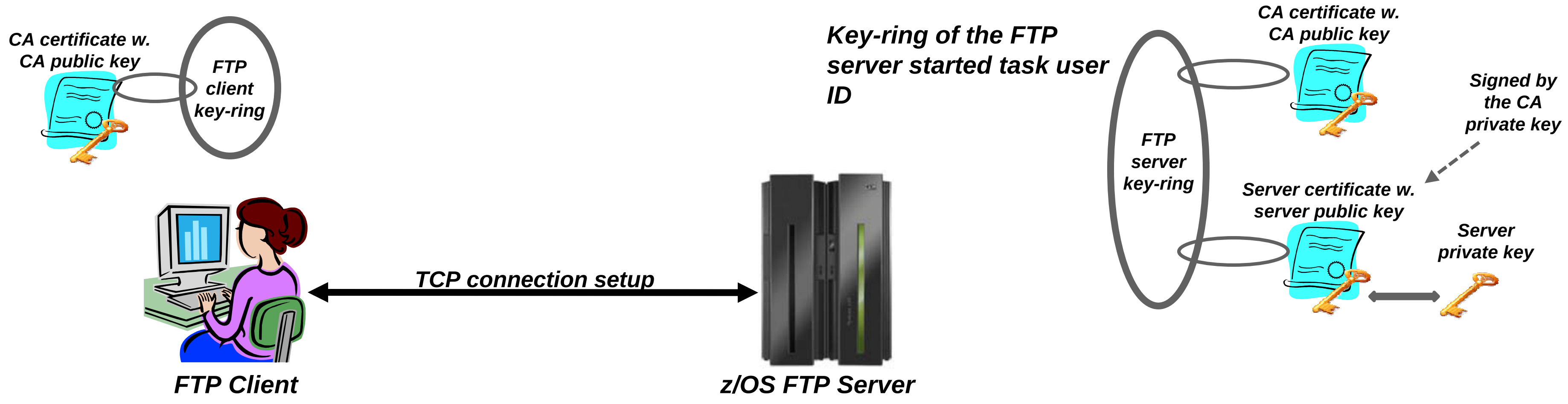
Why it may still fail ..

- **Some firewalls are known to apply various validity checks on the FTP control connection data stream.**
  - One known check is a check to verify that all interactions on the FTP control connection are terminated with an ASCII new-line (NL) character.
  - Most of those checks will fail when the control connection is secured with SSL/TLS since the data is encrypted.
  - If despite following the above guidelines, you run into problems establishing SSL/TLS secure FTP sessions through firewalls, verify with your firewall administrators whether your firewalls implement such checks on the FTP control connection, and consider disabling those checks.
- **Other firewalls are known to disable active mode data connections by default and will block all active mode data connections.**
  - Use passive or extended passive mode FTP instead.
- **Finally, many firewalls monitor activity on TCP connections and will terminate connections that are idle for a certain period of time.**
  - While a large data transfer occurs over an FTP data connection, the FTP control connection is idle.
  - To avoid having firewalls terminate idle FTP connections, consider using the z/OS FTP option FTPKEEPALIVE for the control connection and DATAKEEPALIVE for the data connection.

Safe and Secure Transfers with z/OS FTP

Secure FTP: Keys and certificates overview

What is needed for z/OS Server authentication only



Hello – I want to use SSL/TLS →

← *Hello – OK, me too !!*
And here is my server certificate

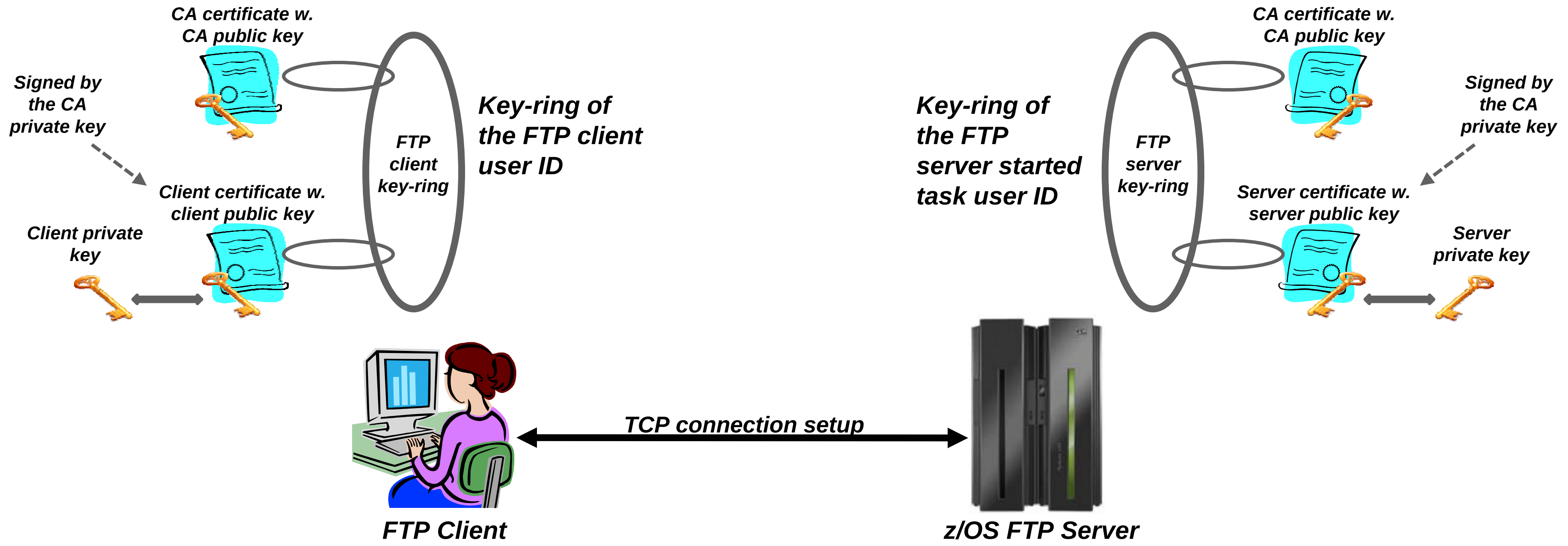
Server certificate w. server public key

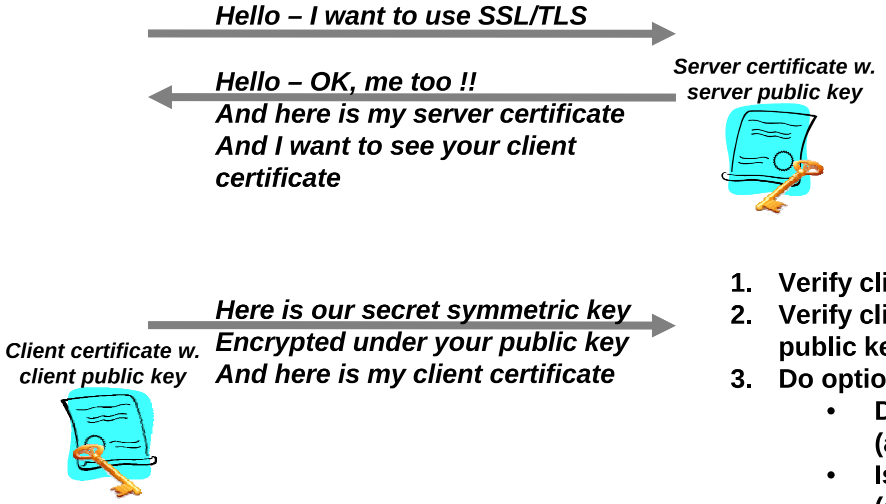
1. **Verify server certificate has not expired**
2. **Verify server certificate is valid using CA's public key**
3. **Do optional checks on the server certificate**
4. **Store server's public key for later use**
5. **Generate symmetric key and encrypt under server's public key**

Here is our secret symmetric key →
Encrypted under your public key

- **CA may be an external CA, such as Verisign, or it may be an in-house CA**
  - In both cases, the CA root certificate needs to be present at both the client and the server side
- **The server certificate is signed by the CA and is stored on the server side**
  - On z/OS, this will typically be the default certificate in the FTP server's started task user ID's key-ring in RACF
- **During SSL handshake, the server certificate (not the server private key) is sent to the client**
  - The client verifies the certificates signature using the CA public key in its copy of the CA certificate

What is needed for z/OS Server and client authentication


CA certificate w. CA public key
Signed by the CA private key
Client certificate w. client public key
Client private key
FTP client key-ring
Key-ring of the FTP client user ID
Key-ring of the FTP server started task user ID
FTP server key-ring
CA certificate w. CA public key
Signed by the CA private key
Server certificate w. server public key
Server private key
TCP connection setup
FTP Client
z/OS FTP Server


Hello – I want to use SSL/TLS
Hello – OK, me too !!
And here is my server certificate
And I want to see your client certificate
Server certificate w. server public key
Here is our secret symmetric key
Encrypted under your public key
And here is my client certificate
Client certificate w. client public key

1. Verify server certificate has not expired
2. Verify server certificate is valid using CA's public key
3. Do optional checks on the server certificate
4. Store server's public key for later use
5. Generate symmetric key and encrypt under server's public key

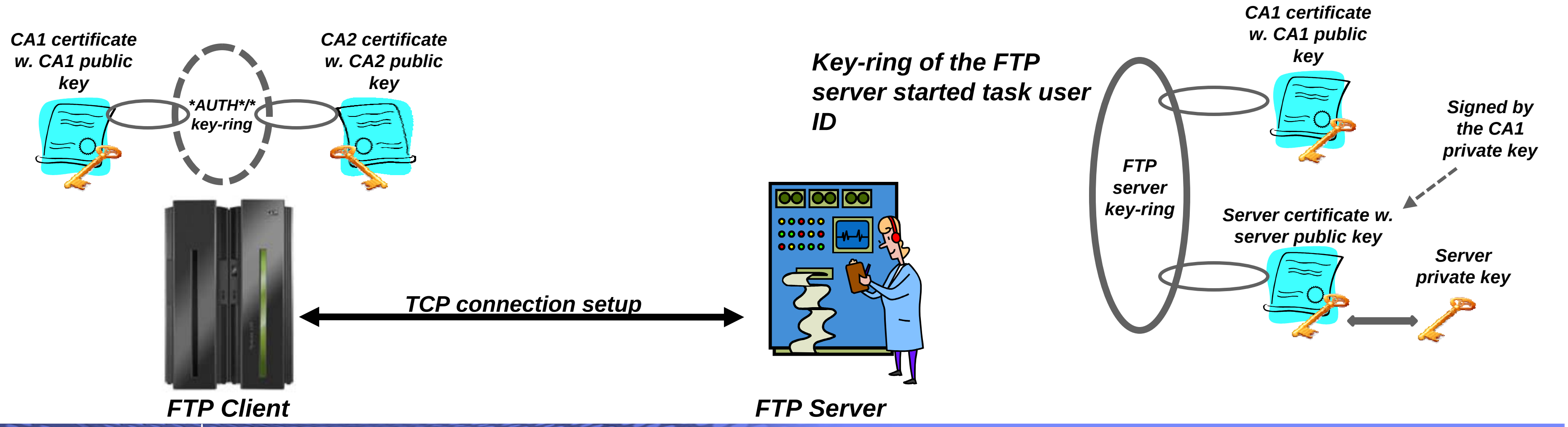
1. Verify client certificate has not expired
2. Verify client certificate is valid using CA's public key
3. Do optional checks on the client certificate
   - Does it map to a RACF user ID (authentication level 2)
   - Is the user permitted to use this service (authentication level 3)

z/OS FTP server options for authenticating an FTP client

Authentication level	FTP server SECURE_LOGIN option	Description
Level 1	REQUIRED	The authenticity and validity of the client certificate is verified against the trusted roots in the FTP server's key-ring.
Level 2	VERIFY_USER	Same as level 1 PLUS a verification that the client certificate is registered by RACF and mapped to a known RACF user ID.
Level 3	VERIFY_USER	Same as level 2 PLUS a verification that the user ID has permission to a SERVAUTH profile that represents this specific FTP server: EZB.FTP.sysname.ftpdaemonname.PORTnnnnn

Virtual key-rings are useful when z/OS is the FTP client

- If z/OS is the FTP client, does every FTP user on z/OS have to have a key-ring with a copy of the CA certificate?
  - A few releases back, the answer was yes
    - What we call an "administratively heavy process"
  - z/OS V1R8 added support for something known as a virtual key-ring
- To have System SSL check all CERTAUTH certificates in RACF when verifying a certificate that was received during the SSL handshake, specify a key-ring in the client FTP.DATA (or matching AT-TLS definitions) as:
  - KEYRING *AUTH*/*
- If client authentication is required, the z/OS FTP user still needs his/her own key-ring


CA1 certificate w. CA1 public key
AUTH/* key-ring
CA2 certificate w. CA2 public key
Key-ring of the FTP server started task user ID
CA1 certificate w. CA1 public key
Signed by the CA1 private key
FTP server key-ring
Server certificate w. server public key
Server private key
TCP connection setup
FTP Client
FTP Server

Safe and Secure Transfers with z/OS FTP

Appendix A:
RACDCERT commands to create keys and certificates for a secure z/OS FTP server

Create self-signed root certificate for test purposes

```
RACDCERT CERTAUTH GENCERT +
         SUBJECTSDN( +
           CN('MVS098 Certificate Authority') +
           OU('Z/OS CS V1R11', 'ENS', 'AIM', 'SWG') +
           O('IBM') +
           L('Raleigh') +
           SP('NC') +
           C('US') ) +
         SIZE(1024) +
         NOTBEFORE(DATE(2009-01-01)) +
         NOTAFTER(DATE(2015-12-31)) +
         WITHLABEL('ABCTLS CA') +
         KEYUSAGE(CERTSIGN) +
         ALTNAME( +
           DOMAIN('mvs098o.tcp.raleigh.ibm.com') )
```

Create a self-signed root certificate and a private/public key-pair:
•CERTAUTH
•KEYUSAGE(CERTSIGN)
•Absence of a SIGNWITH option

- In a production environment, you would not need a self-signed root certificate. To sign server and personal certificates, you would use your company root certificate or an external Certificate Authority.
- For testing, a self-signed root certificate is useful. It allows you to familiarize yourself with keys and certificates and allows you to thoroughly test your secure FTP setup on z/OS before deploying it in production.

Create server certificate signed with your own root certificate

```
RACDCERT ID(TCPCS) GENCERT +
         SUBJECTSDN( +
           CN('MVS098 Server Certificate') +
           OU('Z/OS CS V1R11', 'ENS', 'AIM', 'SWG') +
           O('IBM') +
           L('Raleigh') +
           SP('NC') +
           C('US') ) +
         SIZE(1024) +
         NOTBEFORE(DATE(2009-01-01)) +
         NOTAFTER(DATE(2015-12-31)) +
         WITHLABEL('ABCTLS TCPSERV') +
         KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) +
         ALTNAME( +
           DOMAIN('mvs098o.tcp.raleigh.ibm.com') ) +
         SIGNWITH(CERTAUTH LABEL('ABCTLS CA'))
```

Create a server certificate signed with your own root certificate and a private/public key pair:
•ID(userID) – the started task user ID of your FTP server
•KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)
•SIGNWITH(CERTAUTH LABEL('your rot certificate')

- In a production environment, you would use an alternative procedure after having generated the server key pair and certificate:
  - You would generate a certificate signing request and send it to your CA
  - Your CA would process your request and create a certificate signed with the CA private key
  - You would import the signed certificate into RACF

Alternative: use an external CA to sign your server certificate

```
RACDCERT ID(TCPCS) GENCERT +
         SUBJECTSDN( +
           CN('MVS098 Server Certificate') +
           OU('Z/OS CS V1R11', 'ENS', 'AIM', 'SWG') +
           O('IBM') +
           L('Raleigh') +
           SP('NC') +
           C('US') ) +
         SIZE(1024) +
         NOTBEFORE(DATE(2009-01-01)) +
         NOTAFTER(DATE(2015-12-31)) +
         WITHLABEL('ABCTLS TCPSERV') +
         KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN) +
         ALTNAME( +
           DOMAIN('mvs098o.tcp.raleigh.ibm.com') )
RACDCERT ID(TCPCS) GENREQ (LABEL('ABCTLS TCPSERV')) +
         DSN('USER1.PKITEST.SERVERS.REQ')

(**** delay here while CA processes your request ****)

RACDCERT ID(TCPCS) +
         ADD('USER1.PKITEST.SERVERS.CRT') +
         TRUST +
         WITHLABEL('ABCTLS TCPSERV')
```

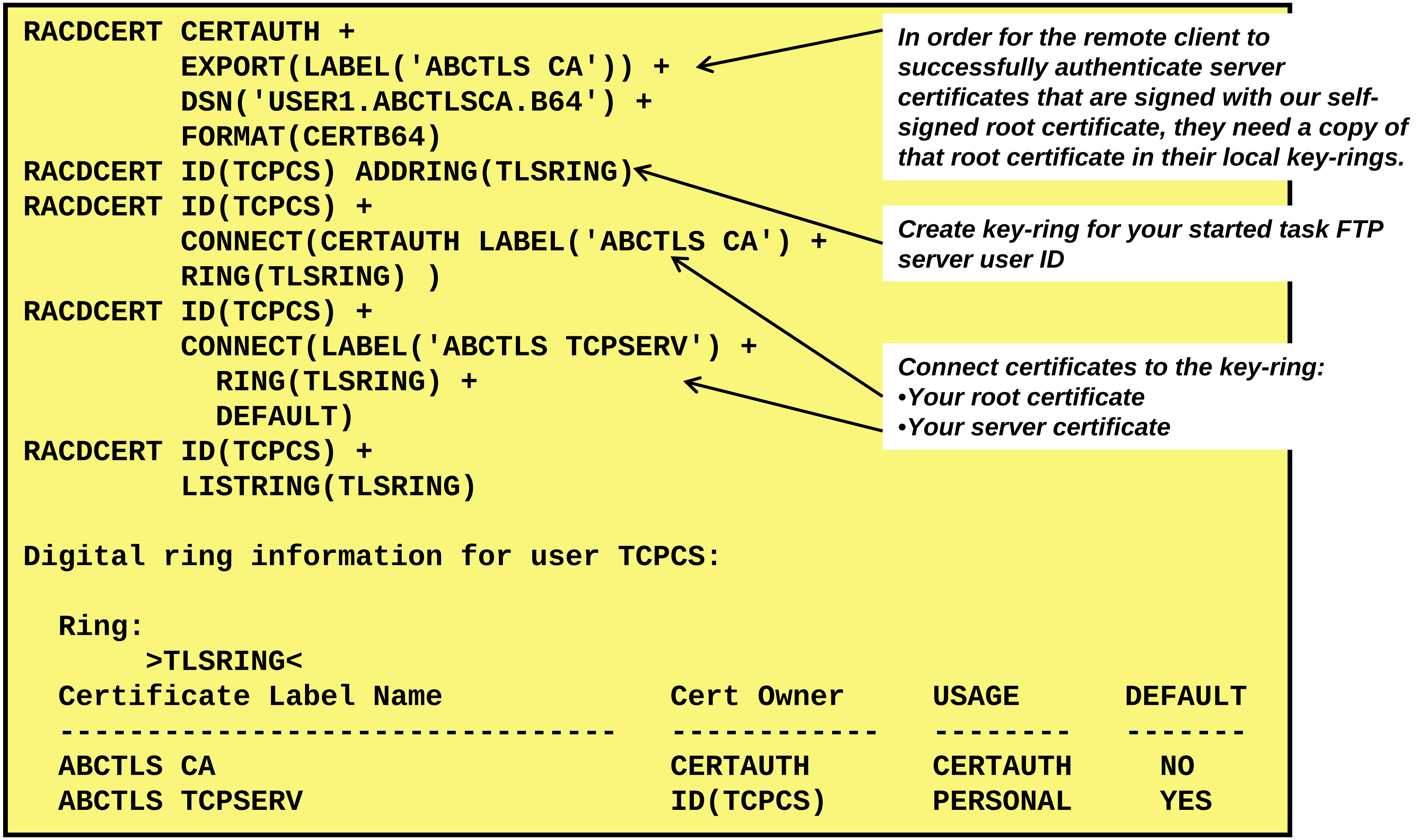
Create a server certificate and a private/public key pair:
- *ID(userID) – the started task user ID of your FTP server*
- *KEYUSAGE(HANDSHAKE DATAENCRYPT DOCSIGN)*

Generate a request to have the cerificate signed by an external CA
- *Send the request to the CA*
- *Receive the response from the CA*

Add the signed certificate into RACF

If not already there, you also need to add the CA's root certificate to RACF as a CERTAUTH certificate

Create you z/OS server started task user ID key-ring and connect required certificates to it



```
RACDCERT CERTAUTH +
         EXPORT(LABEL('ABCTLS CA')) +
         DSN('USER1.ABCTLSCA.B64') +
         FORMAT(CERTB64)
RACDCERT ID(TCPCS) ADDRING(TLSRING)
RACDCERT ID(TCPCS) +
         CONNECT(CERTAUTH LABEL('ABCTLS CA') +
         RING(TLSRING) )
RACDCERT ID(TCPCS) +
         CONNECT(LABEL('ABCTLS TCPSERV') +
           RING(TLSRING) +
           DEFAULT)
RACDCERT ID(TCPCS) +
         LISTRING(TLSRING)

Digital ring information for user TCPCS:

  Ring:
       >TLSRING<
  Certificate Label Name             Cert Owner     USAGE      DEFAULT
  --------------------------------   ------------   --------   -------
  ABCTLS CA                          CERTAUTH       CERTAUTH     NO
  ABCTLS TCPSERV                     ID(TCPCS)      PERSONAL     YES
```

In order for the remote client to successfully authenticate server certificates that are signed with our self-signed root certificate, they need a copy of that root certificate in their local key-rings.

Create key-ring for your started task FTP server user ID

Connect certificates to the key-ring:
•Your root certificate
•Your server certificate

Safe and Secure Transfers with z/OS FTP

Appendix B:
Secure z/OS FTP server FTP.DATA and associated ATTLS policy

z/OS FTP server secure setup example – page 1 of 4

```
;EXTENSIONS         AUTH_GSSAPI        ; Enable Kerberos authentication
                                       ; Default is disabled.

 EXTENSIONS         AUTH_TLS           ; Enable TLS authentication
                                       ; Default is disabled.

 TLSMECHANISM       ATTLS              ; Server-specific or ATTLS
                                       ; ATTLS - use ATTLS
                                       ; FTP - server-specific (D)

 SECURE_FTP         ALLOWED            ; Authentication indicator
                                       ; ALLOWED        (D)
                                       ; REQUIRED

 SECURE_LOGIN       REQUIRED           ; Authorization level indicator
                                       ; for TLS
                                       ; NO_CLIENT_AUTH (D)
                                       ; REQUIRED
                                       ; VERIFY_USER

 SECURE_PASSWORD    REQUIRED           ; REQUIRED (D) - User must enter
                                       ;     password
                                       ; OPTIONAL - User does not have to
                                       ;     enter a password
                                       ; This setting has meaning only
                                       ; for TLS when implementing client
                                       ; certificate authentication
```

Switch between FTP's built-in SSL/TLS support and ATTLS support

Must all connections be secure or just those who wish to be?

Is client authentication required and if so, at what level?

If client authentication is used at level 3 and a user ID can be matched, is a password still required or not?

z/OS FTP server secure setup example – page 2 of 4

```
;SECURE_PASSWORD_KERBEROS  REQUIRED  ; REQUIRED (D) - User must enter
                                     ;     password
                                     ; OPTIONAL - User does not have to
                                     ;     enter a password
                                     ; This setting has meaning only
                                     ; for Kerberos

 SECURE_CTRLCONN   CLEAR             ; Minimum level of security for
                                     ; the control connection
                                     ; CLEAR          (D)
                                     ; SAFE
                                     ; PRIVATE

 SECURE_DATACONN   CLEAR             ; Minimum level of security for
                                     ; the data connection
                                     ; NEVER
                                     ; CLEAR          (D)
                                     ; SAFE
                                     ; PRIVATE

;SECURE_PBSZ       16384             ; Kerberos maximum size of the
                                     ; encoded data blocks
                                     ; Default value is 16384
                                     ; Valid range is 512 through 32768
```

Server's requirement to security of the control connection. Must be set to CLEAR for the server to accept the CCC command

Server's requirement to security of the data connection

z/OS FTP server secure setup example – page 3 of 4

```
; Name of a ciphersuite that can be passed to the partner during
; the TLS handshake. None, some, or all of the following may be
; specified. The number to the far right is the cipherspec id
; that corresponds to the ciphersuite's name.
;
; When using ATTLS, these are controlled via the ATTLS
; Policy
;
;CIPHERSUITE       SSL_NULL_MD5      ; 01
;CIPHERSUITE       SSL_NULL_SHA      ; 02
;CIPHERSUITE       SSL_RC4_MD5_EX    ; 03
;CIPHERSUITE       SSL_RC4_MD5       ; 04
;CIPHERSUITE       SSL_RC4_SHA       ; 05
;CIPHERSUITE       SSL_RC2_MD5_EX    ; 06
;CIPHERSUITE       SSL_3DES_SHA      ; 0A
 CIPHERSUITE       SSL_AES_128_SHA   ; 2F
;CIPHERSUITE       SSL_AES_256_SHA   ; 35
 CIPHERSUITE       SSL_DES_SHA       ; 09
```

Server's required ciphersuites

z/OS FTP server secure setup example – page 4 of 4

```
; When using ATTLS, the keyring is controlled via the
; ATTLS policy
;
 KEYRING           TLSRING           ; Name of the keyring for TLS
                                     ; It can be the name of an hfs
                                     ; file (name starts with /) or
                                     ; a resource name in the security
                                     ; product (e.g., RACF)

;
; When using ATTLS, the TLS timeout value is controlled via the
; ATTLS policy
;
 TLSTIMEOUT        100               ; Maximum time limit between full
                                     ; TLS handshakes to protect data
                                     ; connections
                                     ; Default value is 100 seconds.
                                     ; Valid range is 0 through 86400

 TLSRFCLEVEL       RFC4217           ; Specify what level of RFC 4217,
                                     ; On Securing FTP with TLS, is
                                     ; supported.
                                     ; DRAFT    (D) Internet Draft level
                                     ; RFC4217      RFC level
```

Server's keyring - prefixed with FTPD's started task userID: userID.TLSRING

Is z/OS FTP server to operate at the old draft RFC level for SSL/TLS or the now existing RFC? NB: default is to use draft - you may want to change that!!!!

ATTLS policy for secure FTP server port 4021

Main parts of an ATTLS policy for a secure FTP server on port 4021 (not the complete policy!)

This policy was created using the Configuration Assistant for z/OS Communications Server

```
TTLSRule                          ABC-FTP-server-port-4021~2
{
  LocalAddr                       ALL
  RemoteAddr                      ALL
  LocalPortRangeRef               portR3
  RemotePortRangeRef              portR2
  Direction                       Inbound
  Priority                        254
  TTLSGroupActionRef              gAct1
  TTLSEnvironmentActionRef        eAct1
  TTLSConnectionActionRef         cAct2~ABC-FTP-4021
}
TTLSConnectionAction              cAct2~ABC-FTP-4021
{
  HandshakeRole                   Server
  TTLSCipherParmsRef              cipher1~Default_Ciphers
  TTLSConnectionAdvancedParmsRef  cAdv2~ABC-FTP-4021
  CtraceClearText                 Off
  Trace                           2
}
TTLSConnectionAdvancedParms       cAdv2~ABC-FTP-4021
{
  ApplicationControlled           On
  SecondaryMap                    On
}
TTLSKeyringParms                  keyR~MVS098
{
  Keyring                         TLSRING
}
PortRange                         portR3
{
  Port                            4021
}
```

For more information

URL	Content
http://www.twitter.com/IBM_Commserver twitter	IBM Communications Server Twitter Feed
http://www.facebook.com/IBMCommserver facebook	IBM Communications Server Facebook Fan Page
http://www.ibm.com/systems/z/	IBM System z in general
http://www.ibm.com/systems/z/hardware/networking/	IBM Mainframe System z networking
http://www.ibm.com/software/network/commserver/	IBM Software Communications Server products
http://www.ibm.com/software/network/commserver/zos/	IBM z/OS Communications Server
http://www.ibm.com/software/network/commserver/z_lin/	IBM Communications Server for Linux on System z
http://www.ibm.com/software/network/ccl/	IBM Communication Controller for Linux on System z
http://www.ibm.com/software/network/commserver/library/	IBM Communications Server library
http://www.redbooks.ibm.com	ITSO Redbooks
http://www.ibm.com/software/network/commserver/zos/support/	IBM z/OS Communications Server technical Support – including TechNotes from service
http://www.ibm.com/support/techdocs/atsmastr.nsf/Web/TechDocs	Technical support documentation from Washington Systems Center (techdocs, flashes, presentations, white papers, etc.)
http://www.rfc-editor.org/rfcsearch.html	Request For Comments (RFC)
http://www.ibm.com/systems/z/os/zos/bkserv/	IBM z/OS Internet library – PDF files of all z/OS manuals including Communications Server

For pleasant reading ….