PKWARE®

Technical Whitepaper

Tales From the Frontlines: SecureZIP® and IBM System z™ Integrated Cryptographic Services Facility

By Mike Stebner & Joe Sturonas
PKWARE, Inc.

Table of Contents	Page
SecureZIP® and System z™ ICSF Executive Summary	3
SecureZIP Overview	4
Cryptographic Performance	5
ICSF Operating Environment	5
Cryptographic Hardware for System z	6
Activating ICSF Operations	8
ICSF Application Integration	9
Determining Available ICSF Facilities	9
Application Integration Design Considerations	13
Identify the active ICSF Environment	13
Determine a viable Operating Environment	14
API Selection	15
Integrate all cryptographic aspects	16
RACF® Service Protection	17
Reflections on ICSF Experiences	18
ICSF Performance Considerations	19
Conclusion	22

SecureZIP and System z ICSF Executive Summary

PKWARE leverages the power of IBM's System z Integrated Cryptographic Services Facility (ICSF) with its V9 Mainframe products. PKWARE became familiar with IBM's ICSF while developing SecureZIP V9 for z/OS. SecureZIP provides a cross platform security solution that offers data encryption, digital signing and authentication both outside and within IBM hardware cryptographic environments.

PKWARE's design goal for V9 was to allow SecureZIP to leverage the System z hardware and software facilities, regardless of the hardware features enabled in a specific installation. SecureZIP maximizes the investment made by customers in hardware cryptography by utilizing the least expensive processor capabilities within a system, while maintaining the data security and portability that the ZIP file format provides.

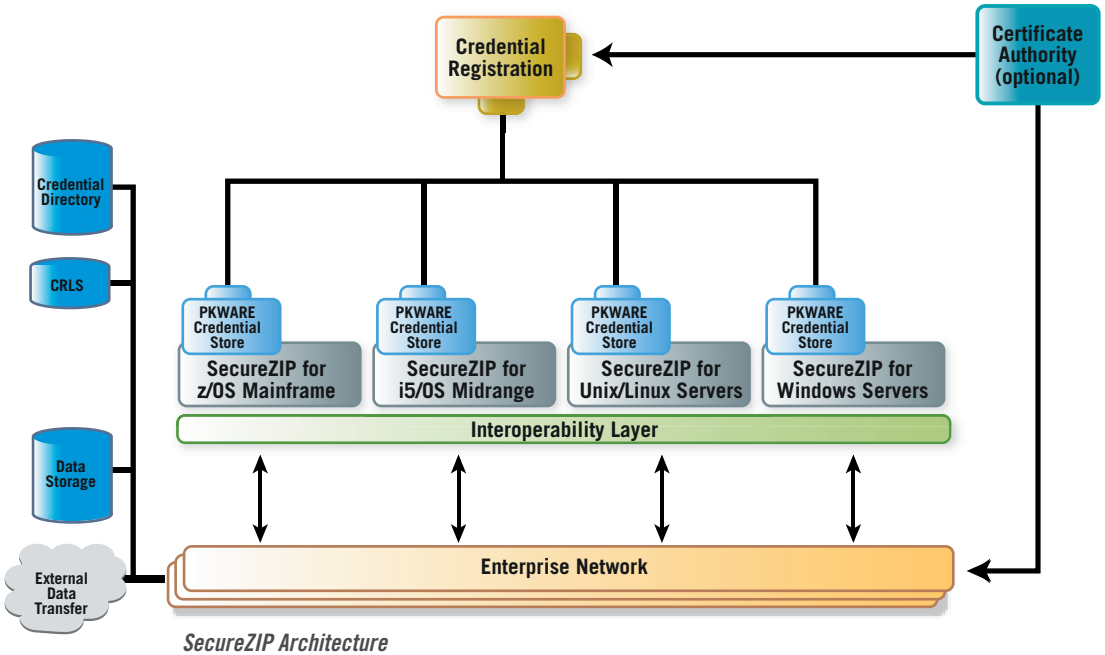
Also included in this whitepaper:

- Key IBM ICSF reference materials
- An overview of the ICSF operating environment (hardware and software scenarios)
- Activating ICSF in a typical System z installation.

Leveraging IBM's ICSF will enable organizations to take advantage of significant cost and resource savings. Learn how you can benefit from these efficiencies through a more detailed technical look at the SecureZIP application usage of ICSF, along with PKWARE's experiences with ICSF API's. This information will be supported by performance comparisons of ICSF processes using SecureZIP.

SecureZIP Overview

SecureZIP is optimized to protect information in transit. The following information will provide some background on PKWARE's motivation to engineer hardware crypto capabilities into SecureZIP for z/OS.



SecureZIP Architecture

SecureZIP adds security to the traditional ZIP file format by compressing and encrypting ZIP archives. As with other PKWARE products, a ZIP archive created with SecureZIP for z/OS can be transferred to IBM midrange i5/OS, UNIX, Linux, Windows Servers and/or Windows desktops. For text files, data transformation is available for EBCDIC to ASCII and vice versa.

SecureZIP for z/OS enables the user to create and extract ZIP archives with enhanced security features, such as the use of passphrases and/or digital certificates to provide strong encryption to ZIP archives. SecureZIP also supports digital signing and authentication. SecureZIP for z/OS supports its own certificate stores, as well as support for LDAP to access public keys.

The certificate store architecture used in SecureZIP for z/OS consists of four stores, the Public Store, Private Store, Certificate Authority Chain Store (Intermediate Root Store) and the Trusted Root CA Store. The Public Store consists of selectable target recipients for certificate-based encryption and selectable authentication certificates. The Private Store consists of decryption certificates and digital signature certificates. The CA Chain or Intermediate Root Store holds the supporting trust chain certificates. The Trusted Root CA Store consists of the final authentication trust point root certificates.

SecureZIP provides portable data security by protecting information once it has been archived as a ZIP file, so in order to maintain its key values, SecureZIP for z/OS had to provide strong, persistent security, while still allowing the protected information to be interoperable with all the other platforms that SecureZIP supports. This illustrates why it was so critical to PKWARE to ensure that any of the crypto that was used on System z could retain its platform interoperability.

Cryptographic Performance

The following information provides an overview of Symmetric encryption, Asymmetric encryption, and a Hybrid Cryptosystem, which is the system that SecureZIP uses.

Symmetric-key algorithms are sometimes known as private-key or single shared-key algorithms. These algorithms represent a shared secret between two or more parties. In other words, the key that is used for encrypting the data is also used for decrypting the data.

Symmetric Key:

Asymmetric key algorithms are known as public key cryptography. Using public key cryptography, there are 2 mathematically related keys, a public key used for encryption, and a private key used for decryption. The public key is generally available to everyone, so that others can encrypt data for that individual. The private key must be kept secret.

Asymmetric Key:

Symmetric key algorithms tend to provide fast encryption, but difficult to share keys. Asymmetric key algorithms are much more secure and computationally intensive than symmetric algorithms, so they tend to have performance issues when processing large amounts of data. A Hybrid Cryptosystem, which is what SecureZIP uses, takes advantage of the strengths of asymmetric and symmetric algorithms.

With a Hybrid Cryptosystem, a symmetric key algorithm is used to encrypt the bulk of the data very quickly. Then using an asymmetric key algorithm, the symmetric key that was initially used to encrypt the data is encrypted. Because the symmetric key itself is a small amount of data, the performance impact is negligible. So a Hybrid Cryptosystem functions like a digital envelope that contains the public key-encrypted symmetric key, while the bulk of the message, which resides within the envelope has been encrypted with a symmetric key.

ICSF Operating Environment

There are many different uses of IBM cryptographic capabilities, but this paper will address just the functionality that was required to implement SecureZIP. For example, IBM Hardware crypto can perform functions that relate to Secure Sockets Layer or SSL, but since SecureZIP for z/OS does not need to interface with SSL applications, those functions were not exploited. The crypto functions that SecureZIP for z/OS did need to exploit included Encryption, Hashing and Pseudo Random Number Generation.

Encryption involves conversion of plaintext into ciphertext. Decryption converts ciphertext into plaintext. Hashing involves using a one-way calculation to condense a long message into a message digest, used in digital signing and authentication. Pseudo Random Number Generation is used in encryption and digital signing.

Not all ICSF functions are available in hardware on all System z platforms. Whether the ICSF functions are available in hardware or software depends on the platform, the hardware crypto cards installed on those platforms, and the specific machine models.

Cryptographic Hardware for System z

This is a representative overview of the cryptographic hardware that is available on IBM mainframe hardware. This does not describe the exhaustive list of cryptographic hardware that is available from IBM on all System z platforms, but rather the most common hardware that PKWARE had to be familiar with in order to integrate ICSF into SecureZIP for z/OS.

The z800 and z900 along with the MP 2000, MP 3000, and 9672 has available to it the Cryptographic Coprocessor Facility or CCF, the PCI Cryptographic Coprocessor or PCICC and the PCI Cryptographic Accelerator or PCICA.

The CCF can have up to two cryptographic coprocessors as high-speed extensions of the central processor. The processor complex can be configured to run in either single-image mode or LPAR mode.

CCF contains sixteen sets of internal keys and working registers, allowing it to be used with multiple LPARs with each LPAR having a different set of master keys. That is, each LPAR appears to see a dedicated set of cryptographic coprocessors. These are known as cryptographic domains and map exactly to LPARs.

The CCFs offer cryptographic services such as DES, TDES, CDMF, MAC Processing, PIN Processing, SHA-1, hardware pseudo random number generator, and Key Management concurrent with asynchronous cryptographic functions such as 1024-bit RSA, and Digital Signature Standard with secure remote master key entry.

The PCICC, which works in conjunction with the CCF, provides the capability of generating and retaining RSA keys in secure hardware. This capability meets a requirement to become a SET (Secure Electronic Transaction) Certificate Authority.

PCI Cryptographic Coprocessor allows up to eight PCICC cards; however, the total number of PCICC and PCICA cards cannot exceed eight. Each card contains two cryptographic processors. It also contains 16 key and working registers, which can support multiple LPARs. The PCICC cards also hold up to 16 sets of DES master keys and PKA or Public Key Algorithm master keys.

In addition to the CCF cryptographic capabilities, the PCICC cards offer RSA Key generation for public/private key pair generation, and 2048-bit RSA signature generation.

The PCICC cards are managed by z/OS ICSF, which provides cryptographic service requests automatically balanced among all available suitable cryptographic engines.

PCICA allows up to six PCICA cards; however, the total number of PCICC and PCICA cards cannot exceed eight. Each card contains two cryptographic processors. PCICAs enable SSL performance enhancement.

The PCICA card is designed specifically for SSL processing, handling any key size up to 2048 bits. It does not support symmetric cryptography. Each processor (there are two per card) can support up to 2100 SSL handshakes per second. The system cryptographic coprocessors and the PCICC cards can also handle SSL handshakes, but not at the same rate. Usage includes: z/OS HTTP server, WebSphere, TN3270 server, LDAP server, and the CICS Transaction Gateway server.

The z890 and z990 supports the PCI XCryptographic Coprocessor or PCIXCC.

PCIXCC is an asynchronous cryptographic coprocessor. This feature is the replacement hardware for most of the secure key functions previously available with the CCF and PCICC hardware features. Both DES/TDES and RSA key functions are supported. For RSA, the maximum key length is 2048-bits.

The z890, z990, z9 EC and z9 BC all support CP Assist for Cryptographic Functions or CPACF and Cryptographic Express2 Coprocessor or CEX2C.

CPACF provides a limited set of DES, TDES and SHA-1 capabilities. The purpose of the CPACF is to provide high performance for encipher and decipher operations and hashing functions.

CPACF is a set of cryptographic instructions available on all CPs. CPACF on the z9 EC and z9 BC machines has been enhanced to include support of the Advanced Encryption Standard (AES) for 128-bit keys along with Secure Hash Algorithm-256 (SHA-256), and Pseudo Random Number Generation (PRNG).

The CPACF can be directly invoked by applications using new machine instructions. Alternatively the CPACF can be invoked using the API provided by ICSF.

CEX2C provides equivalent function to the PCIXCC, but is packaged differently. The CEX2C feature has two PCI-X adapters, and each can be defined as either a Coprocessor or as an Accelerator. The CEX2C supports clear key RSA acceleration.

The Trusted Key Entry (TKE) workstation is an optional feature that provides a basic key management system: master key and operational key entry support. The TKE offers secure local and remote key management providing an authorized person a method for key identification, exchange, separation, update backup and management. The TKE feature is a combination of workstation hardware and software network-connected. The TKE workstation and code level are designed to provide a security-rich, remote, and flexible method of providing master key entry, and to locally and remotely manage cryptographic coprocessor features.

Note: TKE is not the key storage vehicle nor does it perform the cryptographic functions requested by the ICSF APIs. TKE is strictly a more secure way of entering key values into the cryptographic environment rather than using the ICSF TSO panels provided with the base ICSF support. However, all key entry must be completed using the ICSF panels to move the key values from the secure hardware temporary areas to the final storage locations. With a TKE workstation multiple machines and LPARs can be managed remotely.

ICSF Hardware Capabilities

System Type	HW Enabled
MP 2000/3000 9672-G5/6 via CCF	CSNBOWH: SHA-1 CSNBENC: DES/TDES
z800/z900 via CCF	CSNBOWH: SHA-1 CSNBENC: DES/TDES
z890/z990 via CPACF	CSNBOWH: SHA-1 CSNBSYE: DES/TDES
System z9 EC via CPACF	CSNBOWH: SHA-1 CSNBSYE: DES/TDES/AES(128)
System z9 BC via CPACF	CSNBOWH: SHA-1 CSNBSYE: DES/TDES/AES(128)

Non-z/OS hardware, like the MP 3000 and 9672 can support hardware crypto using the CCF, and thus is limited to DES and TDES encryption. The same is true for the z800 and z900.

The z890 and z990 support the same algorithms in hardware, but these are implemented using CPACF. This provides better performance than CCF.

The z9 EC and the z9 BC support all the algorithms identified by their predecessors, with the addition of AES 128bit encryption as well. Like the z890 and z990, these crypto algorithms are implemented in CPACF.

ICSF Software Capabilities

System Type	SW with HW Enabled
MP 2000/3000 9672-G5/6	AES128/192/256 HASH MD5
z800/z900	AES128/192/256 HASH MD5
z890/z990	AES128/192/256 HASH MD5
System z9 EC	AES192/256\ HASH MD5
System z9 BC	AES192/256 HASH MD5

When the algorithm is available in hardware and is active and enabled, ICSF will direct the work to be performed by hardware. In other words, it is not possible to execute an algorithm using software if it is available in hardware. Logically this makes sense and does not present itself to be a problem because if the hardware is available, it should be used.

When hardware is not available, these algorithms are available in software. For example, the z800 and z900 do not support any AES algorithms in hardware, but ICSF supports these algorithms in software.

Activating ICSF Operations

The following provides a summary of steps for preparing a system for hardware crypto.

Even when using CPACF, where crypto functions exist on the CP itself, it still needs to be enabled, so as to comply with crypto export restrictions.

You will need to define and activate the coprocessors to the LPAR's and activate.

The CKDS or Cryptographic Key Data Set is the data storage for the symmetric (DES) keys and the PKDS or Public Key Data Set is the data storage for the public keys. Keys stored in the CKDS and PKDS are not stored in the clear, but are encrypted using EDE or (Encipher/Decipher/Encipher which is an ANSI standard). ICSF uses ISPF panels to administer the cryptographic hardware and keys.

ICSF is the system software (comes as part of the base OS) that provides the interface to the hardware, and must be active to use the hardware. ICSF runs as a started task and provides the key management interfaces, and access to the CKDS and PKDS as well as the API's for applications to invoke the hardware functions.

ICSF also does the routing of work to the hardware resource that will best perform the requested function. For example, if the hardware platform includes a CPACF, a PCICA and a PCIXCC, some DES functions will be routed to the PCIXCC, while other DES functions will be routed to the CPACF, depending on what API is being invoked.

ICSF uses the master key concept to protect cryptographic keys. Master keys, which are stored in secure hardware in the cryptographic feature, are used to encrypt all other keys on the system. All other keys that are encrypted under these master keys are stored outside the protected area of the cryptographic feature. This is an effective way to protect a large number of keys while needing to provide physical security for only a few master keys. The master keys are used only to encipher and decipher keys. Other key-encrypting keys, called transport keys, also encipher and decipher keys and are used to protect cryptographic keys transmitted to other systems. These transport keys, while on the system, are also encrypted under a master key.

ICSF Application Integration

The foundation of developing applications to successfully use ICSF across the generations of System/390 and z/Architecture systems is the planning process of identifying the correct mix of hardware and software components.

The integration occurs across two levels:

1. System; including hardware platform, operating system level, and ICSF level. The combination of these system components enables various types of cryptographic capabilities
2. Programming APIs to achieve a specified set of cryptographic functionality, such as symmetric encryption used by SecureZIP for z/OS

ICSF provides the meeting ground for both of these areas of integration. However, it does not by itself ensure that all facilities are available for use by an application. Additionally, the capabilities and interfaces available from ICSF differ in accordance with the underlying system environment.

To deal with the dynamics of varying system configurations and capabilities, PKWARE's engineering staff created a matrix to provide continuity across the design effort.

Determining Available ICSF Facilities

The first phase of planning for SecureZIP's use of ICSF was to map targeted cryptographic functionality into facilities that are made available across the platform environments.

Cryptographic Service	SS/390	z/800 & z/900	z/890 & z/990	Z9 109/BC/EC
DES/TDES Hardware Acceleration	CCF HCR7703	CCF HCR7704	CPACF HCR7720	CPACF HCR7720
AES ICSF Software	Not available	CCF HCR7706	CPACF HCR7720	CPACF HCR7720
AES128 Hardware Acceleration	Not available	Not available	Not available	CPACF HCR7730
SHA-1 Hardware Acceleration	CCF HCR7703	CCF HCR7704	CPACF HCR7720	CPACF HCR7720
MD5 ICSF Software	CCF HCR7703	CCF HCR7704	CPACF HCR7720	CPACF HCR7720
Pseudo Random Data Generation	CCF & PCICC HCR7703	CCF & PCICC HCR7704	PCIXCC/CEX2C HCR7720	PCIXCC/CEX2C HCR7720

- The left-hand column defines the primary functional ICSF services required for SecureZIP processing. The primary hardware facility available for each required service, CCF or CPACF, is mapped into the matrix for each platform.
- PKWARE researched the required ICSF software level required to access the hardware facility, which are listed by SMP/E FMID. Although an internal binary release level is available from the major ICSF control block, it is not shown here to avoid confusion with operating system release levels.

As the table shows,

- Different hardware facilities within platforms are engaged by ICSF to provide functional cryptographic services. CCF (shown in blue with respective ICSF releases) and CPACF (shown in red) are mutually exclusive hardware-based facilities provided within different platforms.
- The internal release levels of ICSF do not match the operating system some ICSF releases may run within multiple OS's
- Some important ICSF levels for Symmetric encryption include:
  - AES Software capabilities were provided with HCR7706
  - CPACF facilities for the z890 and above require HCR7720. This affected many supported services, whether software or hardware-based.
  - AES Hardware (128-bit only) encryption for the System z9 is provided with HCR7730 (shown in green)

During the research phase of this project, PKWARE found that the levels of ICSF may or may not be distributed with an OS at an appropriate level to meet a functional service requirement.

As shown previously, specific ICSF release levels may be required to meet a specific cryptographic capability within the operating environment.

Operating System	Distributed ICSF Level	Enabled Feature as Used by SecureZIP
OS/390 2.10	HCR7703	Base ICSF for CSNBENC
z/OS 1.2	HCR7704	
z/OS 1.3	HCR7706	CSNBSYE CPACF (z/x90, z/9)
z/OS 1.4	HCR7706 or HCR7708	
z/OS 1.5	HCR7708	
z/OS 1.6	HCR770A	(Upgrade required for AES128 HW)
z/OS 1.7	HCR7720 or HCR7730	CSNBSYE CPACF for DES/3DES CSNBSYE AES128 hardware (z/9)

- ICSF may, or may not be distributed at an appropriate level with z/OS to meet functional capabilities of the hardware environment or application needs. For example, HCR770A was included in the ADCD build of z/OS 1.6, but HCR7720 is required for AES encryption as well as TDES under a CPACF-based system. Therefore, an upgrade of ICSF is required in that environment.
- ICSF software components are fairly well compartmentalized from the OS. Some success was had by transporting ICSF libraries between zOS 1.6 test systems which had different cryptographic hardware, and used successfully for limited execution testing (not recommended for production). However, there were cases of unexpected system failures (such as critical virtual storage issues) when different releases of ICSF were toggled between on the same system.

The design of an application needs to take another time-line facet of the operating system release into consideration.

When choosing APIs, careful attention must be paid both to the level of ICSF, as well as the supporting hardware facilities that will be used.

Symmetric Key Encipher (CSNBSYE and CSNBSYE1)

Table 68. Symmetric Key Encipher required hardware

Server	Required cryptographic hardware	Restrictions
S/390 G5 Enterprise Server S/390 G6 Enterprise Server	Cryptographic Coprocessor Feature	DES keyword is not supported.
IBM @server zSeries 800 IBM @server zSeries 900	Cryptographic Coprocessor Feature	DES keyword is not supported.
IBM @server zSeries 990 IBM @server zSeries 890	CP Assist for Cryptographic Functions	

Composite of zOS 1.6 & 1.7 ICSF Application Programmer's Guides
© Copyright International Business Machines Corporation 1997, 2004. All rights reserved.

As this table from the ICSF Application Programmers Guide for CSNBSYE illustrates, information is provided in the usage section for each API.

- When working with a suite of cryptographic services to meet an overall security objective in an application, be aware that the Required Cryptographic Hardware column may introduce different facilities. For example, this API, used for AES encryption on a z/890 requires CPACF, which is a foundational part of the system. However, the Random Number Generator API CSNBRNG, which may also be required to meet other phases of the encryption process, requires a PCIXCC or CEX2C card. These facilities have additional configuration and operational requirements that are independent of CPACF.
- For some system/service combinations, the reference to required hardware was not fully specified. For example, on the z/800, a reference to CCF is made for Random Number Generation, but the PCICC Cryptographic Unit is the specific device required to use the API. Since this unit can be managed independently from the CCF for DES encryption, operational anomalies can be encountered if not planned for.
- Notice in the third column, that CSNBSYE may support both DES and AES algorithms, but not always for all platforms.
  - On older 390 or z/OS systems, "DES keyword is not supported," is listed because this API supports CPACF instruction sets. Instead, CSNBENC must be used for DES or TDES encryption.
- When designing for the breadth of currently operating environments, historical ICSF & hardware reference documents may be required to provide operational support on older systems.
- The ICSF publications provided with zOS 1.7 dropped the G5/G6 chip system (Multiprise 2000/3000 & 9672) references, while adding the newer System z9 (now awaiting re-branding of System z, EC and BC) from the recent IBM announcement.

At the second layer of integration, geared specifically for application program design, we now have a completed set of APIs to fill out the matrix for target Cryptographic Service points

Cryptographic Service	S/390	z/800 & z/900	z/890 & z/990	Z9 109/BC/EC
DES/TDES Hardware Acceleration	CCF CSNBENC HCR7703	CCF CSNBENC HCR7704	CPACF CSNBSYE HCR7720	CPACF CSNBSYE HCR7720
AES ICSF Software	Not available	CCF CSNBSYE HCR7706	CPACF CSNBSYE HCR7720	CPACF CSNBSYE HCR7720
AES128 Hardware Acceleration	Not available	Not available	Not available	CPACF CSNBSYE HCR7730
SHA-1 Hardware Acceleration	CCF CSNBOWH HCR7703	CCF CSNBOWH HCR7704	CPACF CSNBOWH HCR7720	CPACF CSNBOWH HCR7720
MD5 ICSF Software	CCF CSNBOWH HCR7703	CCF CSNBOWH HCR7704	CPACF CSNBOWH HCR7720	CPACF CSNBOWH HCR7720
Pseudo Random Data Generation	CCF & PCICC CSNBRNG HCR7703	CCF & PCICC CSNBRNG HCR7704	PCIXCC/CEX2C CSNBRNG HCR7720	PCIXCC/CEX2C CSNBRNG HCR7720

ICSF provides several APIs to engage the required cryptographic facilities. With a focus on the symmetric algorithms used by SecureZIP, different APIs may be required in the application to operate successfully across supported platforms.

1. Older systems using CCF with the G5/G6 chipsets (9672, MP 2000/3000), as well as the z800/z900 use the Encipher/Decipher APIs CSNBENC & CSNBDEC (shown in blue) to process DES & TDES encryption.
2. Newer systems with CPACF use a completely new symmetric key API set; CSNBSYE & CSNBSYD (shown in red) to facilitate the use of DES &TDES.
   - It is important to make use of the correct API on the operating hardware platform. The older CSNBENC/CSNBDEC APIs may continue to function on newer systems through specialized accelerator cards that are not built for high-volume data encryption. This has the potential to introduce significant differences in operational controls as well as performance degradation on newer systems.
3. Note that the newer CSNBSYE/CSNBSYD API provides access to both DES and AES algorithms on the CPACF-based systems.
   - However, the application must support using BOTH APIs on z800/z900 systems if DES & AES encryption is required.
4. CSNBOWH (One Way Hash) and CSNBRNG (Random Number Generation) APIs are used for all environments, but may use different underlying facilities.

Application Integration Design Considerations

Based on the information presented thus far, an application intended to run across a progression of platform environments (whether through upgrades or a mixture of live platforms across an enterprise) must have sufficient intelligence to:

1. Identify the active operating environment
2. Determine whether a facility exists to handle the cryptographic requests
3. Select the appropriate API for each service desired, and finally,
4. Integrate a collective of individual cryptographic elements to achieve the target business objective for the application

Identify the active ICSF Environment

With respect to determining what is available in the active operating environment, ICSF provides two status facilities for the program to access and evaluate.

The Cryptographic Communications Vector Table is the main ICSF environmental control block.

The CCVT is built by ICSF initialization and is anchored in low storage through the use of PSA and ECVT

- It remains active even when ICSF is stopped. Through a combination of the CCVT existence and a set of flags within the CCVT, we can determine whether ICSF is in an acceptably active state for use.
- A variety of state information is present; however, the documented programming interface fields are very limited.

The Query Facility (requires an adequate level of ICSF) provides a callable service to obtain similar information.

- An ICSF upgrade may be required to have this facility available
- ICSF must be active, and a check of the CCVT information should be done before attempting to perform the call if there is any likelihood of ICSF being back-leveled on the executing system, otherwise module fetch abends will occur.

Interpretation of the CCVT and CSFIQF state information is critical, although not always obvious from the brief control block descriptions, or externally through operational interfaces.

Determine a viable Operating Environment

To determine whether a viable environment exists to handle the processing needs, we need to get our matrix into the program's operational flow.

A state analysis routine can be written to determine the critical functionality of the ICSF environment. This is crucial for healthy application execution across different platforms or upgraded systems, as well differently configured LPARs.

```
 OSname<z/OS> OS Ver(01) Rel(07) HWclass<Z9>
 ICSF is Active/CCVTACT
 ICSF is at a proper level for CSFIQF
 z/Architecture Hardware Available -Z9
 CSNBSYE (AES) System Capable with ICSF when
available.
 AES Hardware Available -Z9
 CP Assist For Cryptographic Functions
Available
 There are no valid cryptographic units
ACTIVE.
```

In the sample program output shown for a System z9-109 running zOS 1.7 (derived from operating system control blocks), we can see that:

- ICSF is active
- Based on the ICSF version stored in the CCVT, we know that CSFIQF is available for hi-level language service queries
- z/Architecture hardware instruction sets are available (as would also be true for z/800, z/900, z/890 & z/990 systems)
- ICSF is at a high enough level to support the API for the AES algorithm
- The CCVT indicator for AES hardware encryption is ON. However, it is important to know (without help from the CCVT or CSFIQF) that only AES 128-bit key encryption will be done in the hardware on this platform.
- CPACF is active, so DES, TDES and SHA1 hashing will be done in the CP when those services are requested.
- The non-CPACF cryptographic facility units (such as the accelerator2 card) are not currently available for use, so services provided by these units (such as SSL session establishment or random number generation) would not be available.
  - These units have external configuration requirements that are not discernable at the application layer:
    - The devices may not have been defined for the LPAR
    - Master Session keys may not have been entered for this LPAR (as is the case for the System z9 shown here)
    - The units may have been operationally disabled

```
Z800 --------------------- ICSF Coprocessor Management ------------- Row 1 of 4
COMMAND ===> _                                                SCROLL ===> PAGE

 Select the coprocessors to be processed and press ENTER.
 Action characters are: A, D, E, R, and S. See the help panel for details.

    COPROCESSOR      MODULE ID/SERIAL NUMBER                STATUS
    -----------      -----------------------------------    ------
.   C0               04100000000043FD 04100000000043FD      ACTIVE
.   C1               04100000000041A2 04100000000041A2      ACTIVE
.   P00              94E04777                               DEACTIVATED
.   P01              94E04781                               DEACTIVATED
******************************** Bottom of data ********************************
```

```
D M
PROCESSOR STATUS
ID  CPU  CR              SERIAL
00  +      +              03824A2066
01  +      +              13824A2066

+ ONLINE     - OFFLINE     . DOES NOT EXIST
CR       CRYPTO FACILITY
```

As you can see from an operational view, there is very limited information available regarding active facilities.

The z800 configuration shown here demonstrates that a cryptographic unit and its associated services may not be fully available, and yet show "on-line" at the operating system level. Whereas the CCF component used for symmetric DES encryption is active, the PCI Cryptographic Coprocessors have both been temporarily "Disabled" through the ICSF Dialog.

- In this instance, Random Number Generation would not be feasible through the PCICC, yet symmetric hardware-based DES/TDES would be available through the CCF

API Selection

This section will provide a side-by-side comparison of two different platforms using a programmed state matrix. Both operating environments will be running the same release of ICSF, but under different operation system levels.

```
OSname<z/OS> OS Ver(01) Rel(06)
HWclass<Z/X00>
AES Software Only Available -Z/X00

CryptoAPI Facilities     HW   SW
   AES 128 Encryption    ---  SYE
   AES 192 Encryption    ---  SYE
   AES 256 Encryption    ---  SYE

   3DES Encryption       ENC  ---
   DES Encryption        ENC  ---

   SHA1 Hashing          OWH  ---
   MD5 Hashing           ---  OWH
   SHA256 Hashing        ---  ---

   Random Data Gen       RNG  ---
```

```
OSname<z/OS> OS Ver(01) Rel(07)
HWclass<Z9>
AES Hardware Available -Z9
CP Assist Crypto Functions Available
CryptoAPI Facilities     HW   SW
   AES 128 Encryption    SYE  ---
   AES 192 Encryption    ---  SYE
   AES 256 Encryption    ---  SYE

   3DES Encryption       SYE  ---
   DES Encryption        SYE  ---

   SHA1 Hashing          OWH  ---
   MD5 Hashing           ---  OWH
   SHA256 Hashing        ---  OWH

   Random Data Gen       ---  ---
```

First is the profile of a z/800 running z/OS 1.6

- (in green and blue) Software-based AES with CSNBSYE is supported
- Hardware-based DES with CSNBENC (is shown in orange)
- HASHING with One-Way-Hash yields hardware or software based on the algorithm
- Random Number generation is available (with the PCICC units brought on-line)

The second environment is for a System z9 running z/OS 1.7

- The AES-128 line shows that the same programming API would be used on both systems, but ICSF software emulation would be done on the z/800, while the CPACF hardware facility (shown in green) would be used on the System z9. The "Hardware Available" flag from the CCVT or Query Facility would be on to indicate this enablement.
- Software emulation would still be engaged for AES192 & AES256 on both systems
- When DES or TDES operations are required, a change in APIs is necessary. CSNBSYE using CPACF will now be available on the System z9.
- Hashing is available on both systems with CSNBOWH, but the system z9 supports 256-bit Secure Hash (SHA-1 is 160 bits, and Message Digest-5 is 128 bits)
- The Random Number Generator PCICC support hardware was re-activated on the z/800, but the System z9 has not been fully configured, so its cryptographic unit is not available to service Random Number Generation.
  - A supplemental software PRNG routine would need to be used if Random Number generation were required to complete a cryptographic process, such as seeding an initial vector for Cipher-block-chaining.

Once a full system state has been identified, program segments can determine the proper course of action.

Integrate all cryptographic aspects

When programming for symmetric encryption, there is more to account for than an "Algorithm". Rather, consider referring to the suite of associated logic components as an "encryption method".

Key Generation is the entry point for symmetric encryption

- Within the context of the ICSF environment, SecureZIP uses what IBM refers to as "Clear key encryption". This does not mean that the keys are displayed for everyone to see, but rather that the programming environment exposes the key. The key may be derived from a variety of mechanisms, such as a key-derivation routine used with a user supplied password, or internally accessible data. Some questions regarding key generation have to be dealt with, such as:
  - How is the key generated or selected at both ends of the process for encryption and decryption?
  - If a password is to be used, how is it to be transformed into a usable key, and are there character set translation issues to accommodate?
- To prepare ICSF to use application-derived keys, API CSNBCKM may be used to import the "clear keys" for subsequent use by one of the other symmetric encipher or decipher APIs we've already discussed.

The base algorithm is usually the focal point of the method

There are questions to consider here as well:

- Is the algorithm supported on all operating platforms?
- Are there key length restrictions?
- How will the decryption side know which algorithm and key size to use?
- What buffer sizes can be processed through the algorithms and in what form must they be presented?
  - The DES suite operates in 8-byte blocks, whereas the AES suite operates in 16-byte blocks.
  - This does not mean that a call is required per algorithmically defined block, but the total data stream may be required to operate in the modulo of the block size.

Block Padding

Padding deals with the modulo requirements of the algorithm stream. Since the algorithm must work with "even" bytes in accordance to the algorithm block size, the question is how the trailing (or missing) bytes will be filled in (and subsequently verified at the decryption end). ICSF provides "built-in" padding techniques as extensions to specific encryption APIs as described in the Application Programmer's Guide, although not all pad techniques are available for every API. These include:

- 4700-PAD
- ANSI X9.23
- CUSP
- IPS
- Other padding techniques exist outside of the ICSF specifications, and may be required for compatibility with other operating platform interchanges. These must be coded explicitly into the application above the API call.

Regardless of which one is used, application-level padding controls will be required to match the algorithmic requirements at both the encryption and decryption sides.

Cryptographic enhancing modes are available to further protect the encrypted data stream

- For example, Cipher Block Chaining Exclusive-ORs previously encrypted data in the stream with the next portion of the data stream before it is encrypted. This helps to protect against attacks against the middle of the ciphertext.

Ancillary Services and Operating Facilities

- Some API calls have pre-processing call requirements, such as the previously mentioned clear key import prior to using a symmetric key encipher.
- In some cases, the pre-processing call request may require the use of a facility that is not directly required by the base algorithm call. For example, if Random Number Generation is required to create the initial vector (used for the first Exclusive OR of cipher block chaining), but the PCICC (or equivalent cryptographic unit for the platform being used) is not available, then that portion of the process will prevent the encryption flow from being completed.
- The ICSF System Administrator's Guide provides information regarding key management activities:
  - One area of significance involves control of various service calls, which is covered in a section titled **Setting Up Profiles in the CSFSERV General Resource Class**

RACF® Service Protection

1. Define appropriate profiles in the CSFSERV class: RDEFINE CSFSERV service-name UACC(NONE) other-optional-operands Where service-name is one of the following:

…

CSFOWH **One-way hash generate callable service**

CSFOWH1 **One-way hash generate (with ALET) callable service**

2. Give appropriate users (preferably groups) access to the profiles: PERMIT profile-name CLASS(CSFSERV) ID(groupid) ACCESS(READ)

3. When the profiles are ready to be used, ask the RACF security administrator to activate the CSFKEYS class and refresh the in-storage RACF profiles:

SETROPTS CLASSACT(CSFSERV)

SETROPTS RACLIST(CSFSERV) REFRESH"

© Copyright International Business Machines Corporation 1997, 2004. All rights reserved.

Above is a sample list of action steps for a security administrator to take when establishing protection against unauthorized use of cryptographic facilities.

- Under the first step, the focus is on the One Way Hash APIs, but there are many APIs which induce a call to SAF for a RACF resource check.
- CSNBSYE and CSNBSYD are not checked via SAF & RACF. So, if these, or other unlisted APIs need to be protected from unauthorized use, the application may want to invoke a SAF-enabled API as a pre-call mechanism, or construct supplemental installation-dependent resource rules.
- Pay close attention to the names, because the RACF service name may not be an identical match to the API name being invoked.

Reflections on ICSF Experiences

During the process of integrating ICSF into SecureZIP, PKWARE ran across several key issues

Nomenclature:

Gaining an understanding of the acronyms & terminology for hardware and software encryption components was the first hurdle.

- A combination of hardware manuals, ICSF manuals (over 1,300 pages), White papers & ancillary Redbooks from i5/OS® helped build the vocabulary.

Hardware & ICSF release level requirements:

Determining which facilities are available for use on a system, with various configurations of hardware, OS and ICSF was the next challenge.

- Building the matrix to reflect various components and functionality was critical
- Having regular communication amongst the development team with the matrix as a core reference point was a highly successful best practice

ICSF state indicators:

ICSF has a longer history than current events would reflect.

- As a result, there are a plethora of status indicators (not well documented) to wade through in order to assess the viability of a functional environment.
- Writing prototype utility programs with functions that eventually became a part of the internal programming decision table were useful when assessing ICSF operational states on different platforms.

Changing API functionality & formats:

API functionality and formats have progressed over time, especially with newer hardware technologies that are available on the zArchitecture platforms

- Some older cryptographic functions are still referenced, but are not a strategic part of newer systems
- Some tributaries were followed because they appeared to have the potential for achieving functional objectives, but dried up because they were based on antiquated technologies (not likely to be continued)

Examples for older algorithms:

In some cases it was difficult to locate good programming examples for API calls. i5/OS Redbooks were consulted to locate examples for the coding of older APIs (which are the same on both platforms) because the samples were better documented.

Interpreting return information with reason codes:

There are roughly 30 pages of error reason codes documented in the Application Programmers Guide.

- An initial approach to dealing with errors is to target anticipated failure conditions and test the detection/recovery logic in the programs, such as stopping ICSF or taking underlying hardware facilities off-line
- It also helps to build in an application interpretation of error conditions and translate them into actionable operational constructs (such as a clear text message). Be sure to externalize unknown error reason codes in messages for further analysis.
- Some unexpected errors not directly attributed to ICSF were encountered– Abend 0E0 during an internal SAF call, leading to RACF maintenance

Discerning which facility is required or used:

The use of an intelligent API has its pros and cons

The masking of the facilities being used through ICSF may cause a degradation of performance or mask the use of unintended facilities.

- Sometimes a comparative elapsed time or CPU consumption value for a known workload is the only means of determining that hardware is actually being used.
- For example, an application is being activated on multiple LPARs, or ICSF is being upgraded on multiple systems. If an inadequate release of ICSF is running on a System z9, then AES128 will be running in software emulation mode, not hardware. If an installation has a security policy in place that states hardware encryption must be used, then that policy will be broken without any discernment by the application.

ICSF Performance Considerations

This section illustrates an approach that was taken to assess CPU and elapsed time processing requirements when ICSF cryptographic services are engaged for different platform configurations.

Specific symmetric key encryption and hashing algorithms were selected for comparison purposes across multiple platform and ICSF software configurations.

Although analyzed during the project, Random Number Generation is not presented in this section for the sake of brevity. However, it is important to note that this particular service revealed that there are significant performance differences for various API services, and that all service points should be taken into consideration when assessing total application throughput.

PKWARE reviewed the performance of supported SecureZIP algorithms across the predominant z/OS platforms, including the new System z9. A driver program was written in C to select the appropriate API for each target algorithm based on the selection matrix.

The model for the evaluation was to assess bulk stream data cryptography with 100megabytes passed in 32K buffers per API call.

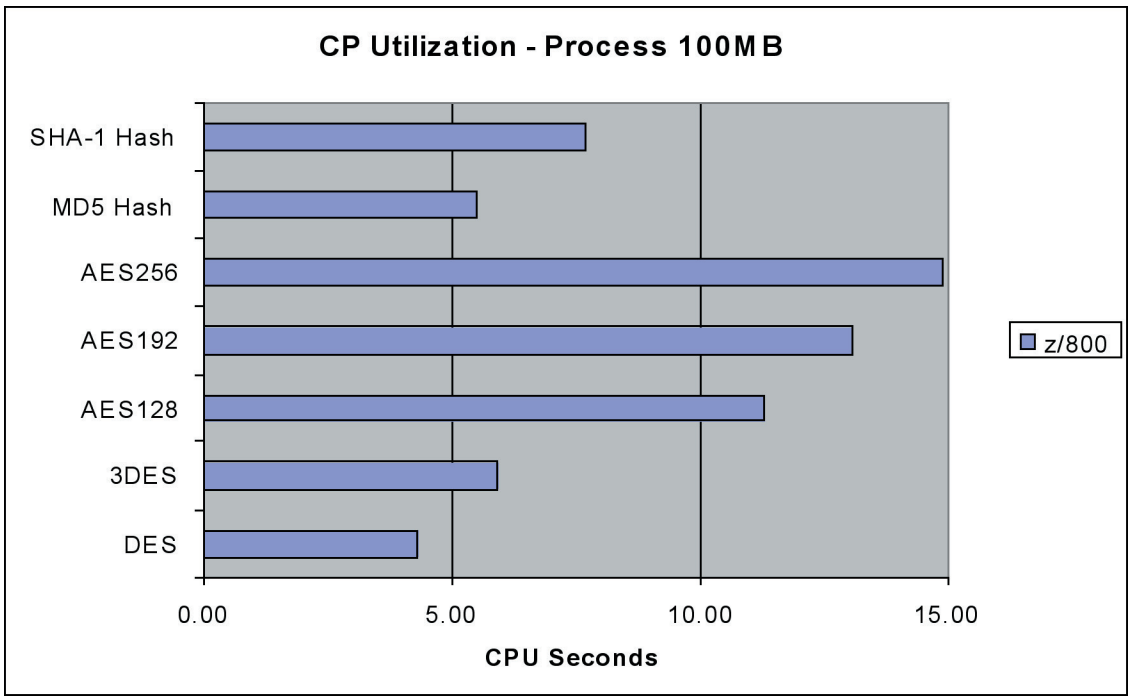
- A single-task program was used to loop through predefined data buffers with minimal application-layer interference
- A single key generation and import was performed prior to passing the data through the algorithm.

System	Config.	MIPS per CP
z/800	Native LPAR in SYSPLEX	132
z/890	Native LPAR	367
System z9-109	VM LPAR	608

The basic configuration of the systems is shown in the table.

- Although the z/800 had 2 CPs active, the model was designed to operate on 1 CP. Execution monitoring also showed that one CP was dedicated during the execution.
- No evaluation was done with respect to z/OS running under z/VM on the System z9

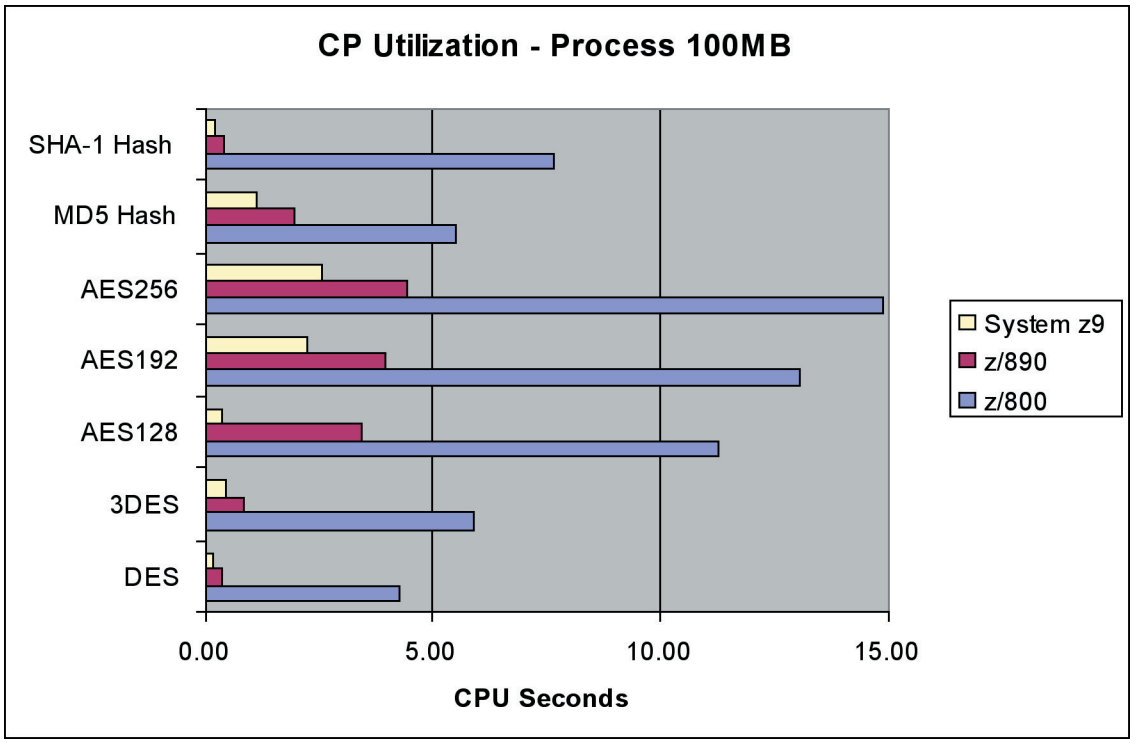
The first graph illustrates the CPU processing requirements on the z/800 with CCF.



In this system configuration, there are a few noteworthy points:

1. The CCF-based hardware hash for 160-bit SHA1 uses more CPU than the software emulated 128-bit Message Digest-5 algorithm
2. The ICSF software-based AES emulation provides newer cryptographic functionality, but takes more resources on this system than the CCF-based hardware TDES. In fact, TDES hardware used a little over half of what AES128 used, and only 45% of the CPU time that AES192 required running with software.

This side-by-side comparison of the same algorithms across the different platforms shows the advances made in hardware-based cryptography. For the algorithms being handled by ICSF software emulation on all systems, such as MD5, AES192 & AES256 the faster processor speeds are evident.



Notice that CPACF has significantly improved SHA-1 hardware hashing over MD-5 software emulation

The relative cost of performing TDES hardware encryption vs. software emulation decreases for the more advanced systems.

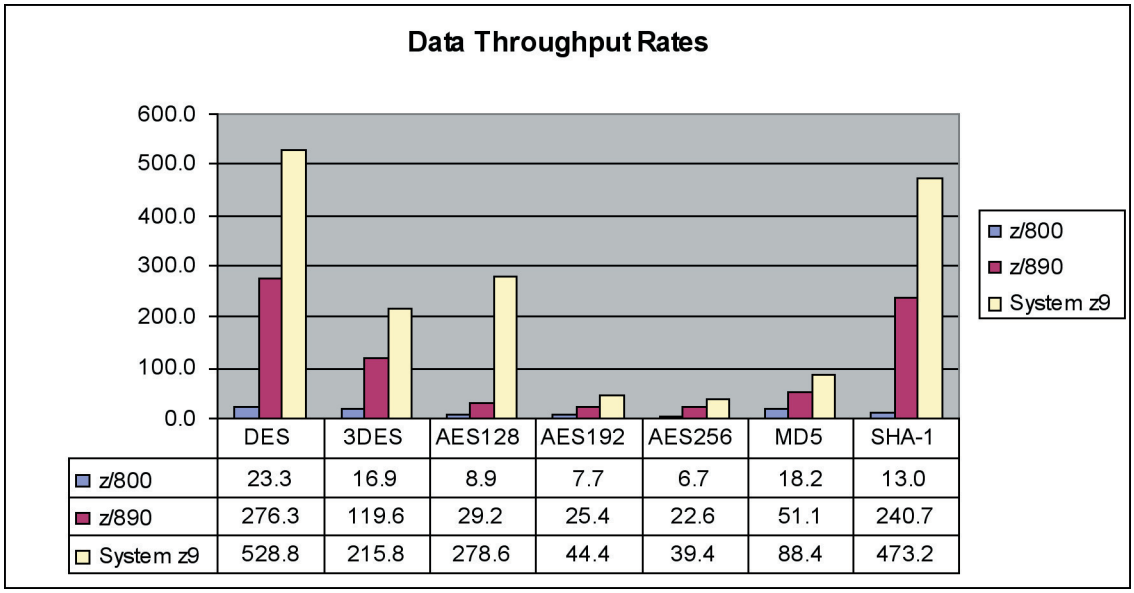
- For example, when comparing TDES hardware encryption against AES192 (which is software-based on all platforms) within its own environment,
- z/800 CCF performs TDES with 45% of the CPU required for AES192
- z/890 CPACF performs TDES with 21% of the CPU
- System z9 CPACF performs TDES with 21% of the CPU

A significant reduction in processor resources comes into play with the System z9 and AES128 Hardware-based encryption

- AES128 with CPACF performs a role reversal and performed about 29% better than TDES.
- However, if you recall the earlier discussion regarding ICSF release levels, if the AES128-CPACF enabled release of ICSF (HCR7730) is not running, then AES128 emulation will be in effect, and the performance will be somewhat worse than TDES.

By inverting the CPU timings in relation to the total amount of data processed, resulting throughput rates for each system can be computed.

- Note: Although 56-bit DES encryption is shown here with a very high rate, in the world of strong encryption, TDES and hardware-based AES128 are clearly the winners.



	DES	3DES	AES128	AES192	AES256	MD5	SHA-1
z/800	23.3	16.9	8.9	7.7	6.7	18.2	13.0
z/890	276.3	119.6	29.2	25.4	22.6	51.1	240.7
System z9	528.8	215.8	278.6	44.4	39.4	88.4	473.2

- This information can be incorporated into the application's overall processing model to project what the end-to-end performance will be.
- The choice of algorithm is then based on an installation's security requirements and available facilities
- There is one other important consideration regarding performance. Additional load tests should be performed when multiple concurrent cryptographic application processes are to be run.

Conclusion

The design objectives of this project were to support IBM Hardware Crypto seamlessly, without impacting any of the interoperability functionality that SecureZIP offers today. PKWARE was effectively able to create a compressed, encrypted ZIP archive using SecureZIP for z/OS and extract the files using SecureZIP on all other platforms.

In addition to underlying the multi-platform deployment of SecureZIP, this functionality is key to our PartnerLink product, where by a Sponsor using SecureZIP for z/OS could encrypt reports that are sent to multiple partners. PartnerLink enables the bi-directional exchange of secure information from a "Sponsor" to their Partners, at no cost to the partner.

The partners could have System i, UNIX, Linux or Windows Server systems, but with a copy of the partner software, SecureLink on one of those platforms, they would be able to decrypt the reports, and also encrypt files back to that sponsor as well. The ability to communicate to any computing platform was the key reason that IBM ICSF integration was so critical to PKWARE. PKWARE enables the faster delivery of shared files, regardless of the operating platform.

Copyright© 2006 PKWARE, Inc. and its licensors. All rights reserved. SecureZIP and PKZIP are registered trademarks of PKWARE, Inc. Trademarks of other companies mentioned in this documentation appear for identification purposes only and are property of their respective companies.

062606

About PKWARE, Inc.

PKWARE is the creator and continuing innovator of the most widely used portable file handling system in the world. Even after 20-years, it continues to be the best available method to efficiently transfer, protect and share information across computing platforms and security infrastructures. PKWARE's ZIP family of products can scale effectively from a single user to the worlds' largest data centers. PKWARE developed PKZIP to help people share information cost effectively when bandwidth was the biggest challenge. To meet the needs of the current market, we've extended our products to enable companies to securely share information both internally and externally. In fact PartnerLink, a special deployment of SecureZIP is optimized to enable information to be shared securely within Enterprise defined "communities of trust." PKWARE empowers users to better manage and share information.

PKWARE®

www.pkware.com

United States
648 N. Plankinton Ave., Suite 220
Milwaukee WI 53203
Phone: 1-888-4-PKWARE
Fax: 1-414-289-9789
info@pkware.com

International
Siena Court, The Broadway
Maidenhead, Berkshire SL6 1NJ
Phone: +44 1628 509019
Fax: +44 1628 509109
internationalmarketing@pkware.com