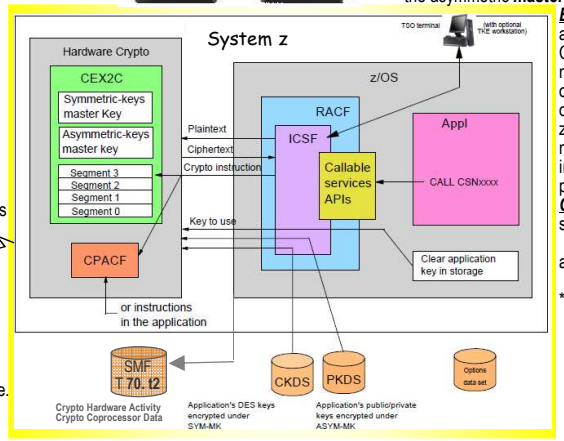
CheatSheet #60 zTidBits System z Crypto & Hardware

* **System z cryptographic hardware** supports four cryptographic capabilities:
  - **Data confidentiality** (encrypting/decrypting data using symmetric and/or asymmetric algorithms)
  - **Message integrity** (message authentication, modification detection, nonrepudiation)
  - **Financial functions** (using symmetric algorithms to protect PINs associated with credit cards or financial transaction)
  - **Key management** (security and integrity of keys)

Clear Key vs *Secure Key* vs *Protected Key*

* IBM Crypto hardware can use either clear key, secure key or protected key and no matter which you choose, there is *no difference* in the *crypto algorithms;* the resulting cipher text is the same if the underlying key is the same.
  NOTE: The difference *is* the protection provided for the *key value* that protects the data.
* The secure hardware includes tamper detecting technology to protect against attacks involving probe penetration, power sequencing, radiation and temperature manipulation consistent with FIPS[1] requirements.
  - If a tamper is detected the circuitry will "**zeroize**" the card wiping out the keys so they cannot be compromised.
  - When a **secure key** is created, that key is encrypted under a **master key**, and the underlying key value is never exposed in the clear, outside of the secure hardware.
  - When that key needs to leave the secure hardware (for example to be stored in a dataset like the CKDS[2]) the encrypted version of the key is stored.
* That encrypted key must itself be decrypted from under the **master key** before it can be used to encrypt or decrypt data.
  NOTE: For transport purposes, a key can be encrypted under a key-encrypting key instead of the **master key**, but in no instance will a secure key exist in the clear outside of the secure hardware.
* A **clear key** may exist on the network during the key entry process, or it may exist in an address space during key entry or when in use by an application.
* With the November, 2009 LIC, the z10 now supports **protected keys** that does *not* rely on the tamper resistant secure hardware, but the protected key is encrypted under a wrapping key[3] until it is used within the CPACF[4] which provides better performance than the secure key hardware. LPAR
  - A protected key is an operational key that is encrypted under a wrapping key associated with the
  - When the protected key is brought into the CPACF it is unwrapped and the clear value is used to perform the crypto operation.
    NOTE: Protected keys can be DES (**Data Encryption Standard**) , TDES (**triple DES**)[5] ,or AES (**Advanced Encryption Standard**) .
*A **secure key** never exists in the clear outside the tamper resistant boundary of the secure hardware
  - A protected key never exists in the clear in system storage.
  - A secure key is encrypted under a secure **master key** when it leaves the tamper resistant hardware.
  - A protected key is encrypted under a wrapping key that is *uniquely created* each time the LPAR is activated or reset through the HMC or Support Element (SE).
    > That wrapping key is stored in the Hardware Storage Area (HSA) -> *See #30 zTidBits (Free Storage – more WAS).*
    NOTE: Access to the HSA is limited to authorized programs and the CPACF wrapping key is not available in operating system or application storage, but it does not have the protection of the tamper resistant hardware technology.
  -There are two variations of the wrapping key, one for **DES/TDES** keys and one for **AES** keys.
* There are additional costs associated with secure keys such as performance and CPU costs, so customers must make a business decision about whether their security requirements warrant the cost of secure key support.
* **Clear keys** provide the best performance of the three types because their operations are done on the general purpose CPs at machine speeds.
* Secure key operations are routed out to the PCI[6] card on the Self-Timed Interface[7], so effectively you're executing an I/O operation to get the data and keys out to the card.
* While the crypto work is offloaded to the card, there is still some CPU costs in getting the work formatted for and routed to the card and then in receiving the results back from the card and passing those results back to the caller.
  NOTE: The real impact on performance is the asynchronous operation to the card where secure key operations will take longer than **clear key** operations.
* Protected keys fall in between **clear keys** and secure keys in terms of performance.
  - Performance is closer to that of **clear keys**, although they do have some additional overhead.
  - It is expected that most protected keys will be stored as secure keys in the CKDS.
    > That key will need to be brought into the Crypto Express3 (CEX3), decrypted from under the **master key** and then re-encrypted under the wrapping key. [*CEX3 same as CEX2, but improved performance and availability*]
    > The actual encryption and decryption of the data, done on the CPACF, will then have similar performance characteristics to a **clear key** operation.
* **Clear keys** would be *appropriate when* there are procedures in place to protect the key values while they are in use, or when the additional cost of secure key protection outweighs the risk to the data (as when a short-lived key is required, such as System SSL).
  - If the amount and/or value of the data being protected is low, the additional cost of secure key protection may not be worthwhile.
  - If operational procedures protect the **clear key** values (i.e. dumps that might contain passwords are shredded or disposed in a secure manner), then secure key encryption may not be required.
  - If the data being protected is "segmented" in such a way that an attacker would have to invest significantly to capture all the relevant pieces, **clear key** may provide sufficient protection.
* **Protected keys** would be appropriate for applications that require better performance than secure key, but don't have the strict requirement of a Hardware Security Module.
  - Protected keys can begin life as a secure key or a **clear key**, that is, a protected key may use a secure key which is decrypted from under the **master key** and then wrapped for use in the CPACF.
  - Or, a protected key may be generated as a **clear key** within an application and then wrapped for use in the CPACF.
* Secure key hardware requires that a **master key** be loaded to enable that hardware.
  - Since it provides protection for other keys, the **master key** must be available before the hardware is enabled.
  - **Clear key** hardware does not require a **master key**.
    Note that secure key hardware may be a superset of **clear key** hardware. That is, **clear key** work may be performed on secure hardware, but secure key work will ***only*** be executed on secure key hardware.



The ICSF callable services comply with the IBM Common Cryptographic Architecture (International Organization for Standardization (ISO) standard 7498-2

1. Federal Information Processing Standard
2. Cryptographic Key Data Set
3. Wrapping Key is a class of symmetric encryption algorithms designed to encapsulate (encrypt) cryptographic key material. The Key Wrap algorithms are intended for applications such as (a) protecting keys while in untrusted storage, or (b) transmitting keys over untrusted communications networks.
4. CP Assist for Cryptographic Function
5. TDES It is so named because it applies the Data Encryption Standard (DES) cipher algorithm three times to each data block.
6. Peripheral Component Interconnect), an industry-standard bus for attaching peripherals to computers.
7. STI -A high-speed interface designed for interconnection of the early System z and S390 I/O subsystem to the processor.

REF: TechDocID: WP100810 & SG24-7470

* **Symmetric keys** are used with symmetric algorithms (Data Encryption Standard or **DES**; Triple DES or **TDES** and Advanced Encryption Standard or **AES**) and *both* parties (the encrypter and decrypter) must have a copy of the key, which must be a secret between the two.
  - Anyone, and everyone, who has a copy of the symmetric key can decrypt the data enciphered with that key.
* **Asymmetric algorithms** or **Public Key Architecture** (**PKA**) use a key pair to protect data.
  NOTE: Don't confuse **clear key** / secure key with public/private keys used by asymmetric algorithms.
  - With PKA, two different, but mathematically related keys are used.
    > The **public key**, is made available publicly and can be used by *anyone* who wants to send data securely to the owner of the **private key**.
    > Data that has been encrypted using a **public key** can ONLY be decrypted using the corresponding **private key**.
    > Anyone who has a copy of the **public key** can encrypt their own data to send, but they *cannot* use that public key to decrypt data that was encrypted using that **public key**.
    > Since the **private key** must be used to decrypt the data encrypted by the **public key**, that **private key** should be well protected and only available to the owner of the public / private key pair where the secure hardware along with the asymmetric **master key** can provide that level of protection.

Export Restrictions The U.S. Government considers encryption technology to be a munitions, and therefore strictly controls the ability to export the technology. Since the System z includes Crypto technology within the machine, IBM controls access to the crypto hardware via microcode, and the U.S. Government limits where that microcode can be exported. All of the System z crypto hardware requires the appropriate microcode to be installed and operational before the export restricted functions can be used. On the z890/z990, z9 and z10 machines, this microcode is ordered as no-charge Feature Code #3863. On the CCF machines, the microcode was unique to the CCF on the machine. IBM software that implements encryption will also check for the presence of this feature code before performing encryption in software.

Cryptographic Software The Integrated Cryptographic Service Facility, **ICSF**, is the system software that provides the interface to the hardware. As new functions are implemented in the hardware, new versions of ICSF will be available to invoke those functions. ICSF is available as a component of z/OS, however the most current versions are available via web download at http://www.ibm.com/servers/eserver/zseries/zos/downloads/.

* **Crypto APIs**: There are a few crypto instructions that are available directly to an application, but most of the crypto hardware can only be accessed by using the cryptographic Application Programming Interfaces (APIs).
  - Invoking the APIs in application code or in a product will pass the crypto request to ICSF which will determine what hardware is available and which device can best service the request.
  - Some APIs may be supported by a single crypto device which implies that that device must be installed and available to service the API.
  - Other APIs can be serviced on several different devices and ICSF will make the decision where best to route each call.
  - The ICSF **Application Programmer's Guide** provides a table for each API that describes the hardware required to support the API located on RESOURCE LINK.
  - ICSF callable services can be called from application programs written in a number of high-level languages as well as assembler.
    [The high-level languages are: C, COBOL, FORTRAN, PL/I] .
* How the application is coded can also affect how the cryptographic hardware is used.
  - The application controls which APIs or cryptographic instructions are invoked, and what parameters are passed via the API.
  - Specific parameters may be supported on a particular cryptographic hardware device, but not on another and therefore, the parameters can impact how the work is routed.
  - Some applications may perform the cryptographic functions in their own address spaces, never routing the work to the hardware or ICSF.
  NOTE: Invoking the API can be *expensive*, some IBM software will only invoke ICSF and the hardware if the amount of data to be encrypted is large enough to overcome the overhead of passing the data to ICSF.
* Beginning with the z890/z990, IBM changed the crypto architecture to move much of the functionality outboard using the **PCI** (Peripheral Component Interconnect) bus.

Several reasons to move crypto function off the system board and into the I/O cage:

1. **Availability**: adding crypto functionality to a processor requires an outage of the entire machin PCI cards are **hot-pluggable**, and so new function could be added by simply plugging the card with the new functionality into the I/O cage.
2. **Scalability**: depending on the platform and other cards installed in the I/O cage, up to eight cryptographic features can be installed in a processor. As workload increases, additional features can be ordered and installed, without an outage to the LPAR.
3. **Cost:** Moving the secure hardware support (the ability to detect probes and physical attacks) to the PCI cards allowed IBM to better manage engineering cost of this additional protection.

* The z10 GA3 (10/09) included support for a *new PCI card, the Crypto Express3* and support in the CPACF for the new protected key function supporting several new crypto functions as well as making it easier to manage the crypto devices.
  - There are **three** crypto hardware devices available on the z10: the CP Assist for Cryptographic Function (CPACF) is integrated into the PU in the host, and the Crypto Express2 and Crypto Express3 are PCI devices installed in the I/O cage.
  NOTE: On the z10 (and the z9), a crypto engine can be *dynamically* changed (from accelerator to coprocessor or coprocessor to accelerator) via the Hardware Management Console (HMC) without taking an outage of ICSF, the operating system or the LPAR. This means that as workload changes you can reconfigure the crypto devices to take advantage of the performance and throughput characteristics of each without incurring an outage.