A Clear Key/Secure Key/Protected Key Primer

Abstract: Beginning with the latest updates to the z10 microcode and the new support in ICSF, FMID HCR7770, IBM cryptographic hardware supports three types of keys. This paper describes the basic differences between clear key, secure key and protected key, and is an introduction to how the hardware provides additional protection for secure keys. Understanding the difference between the three will help in designing the right cryptographic solutions and in determining the hardware requirements for the cryptographic work.

Encryption is the process of scrambling data, in order to protect it. The data is scrambled using a cryptographic algorithm (a series of steps), controlled by a key. A key is a sequence of binary digits that is input to the algorithm. The security of encryption relies upon keeping the value of the key a secret.

In cryptography, all symmetric keys and the private key of a public/private key pair must be secured to protect the data. For symmetric keys, the key value needs to be protected so that only the two parties exchanging the encrypted data know the value of the keys. The DES, TDES and AES algorithms are published, so the key provides the security, not the algorithm. If a third party has access to the key, they can recover the data just as easily as the intended recipient.

For asymmetric keys, the private key must be protected so that only the owner of the public/private key pair has access to that private key. The public key can and will be shared with partners who will send encrypted data to the owner of the keypair.

When a key is defined in the System z crypto environment as a secure key, the key is protected by another key called a master key. IBM secure key hardware provides a tamper-sensing and tamper-responding environment that, when attacked, will zeroize the hardware and prevent the key values from being compromised. The secure key hardware requires that a master key be loaded. That master key is stored inside the secure hardware and used to protect operational keys. The clear value of a secure key is generated inside the hardware (via a random number generator function), and encrypted under the master key. When a secure key must leave the secure hardware boundary (to be stored in a dataset) that key will be encrypted under the master key. So the encrypted value is stored, not the clear value of the key. Some time later, when data needs to be recovered (decrypted), the secure key value will be loaded back into the secure hardware, where it will be decrypted from under the master key. The original key value will then be used, inside the secure hardware, to decrypt the data. If the secure key is stored in the CKDS, and the master key changes, ICSF provides the ability to re-encipher the secure key; that is to decrypt it from under the original master key and re-encrypt it under the new master key, all within the secure hardware, before it is stored back into a new CKDS, now associated with the new master key value.

A secure key may also be encrypted under a key-encrypting-key or transport key, when it needs to be shared with a partner. In this case, it will be encrypted under the transport key, not the master key, when it leaves the secure boundary of the hardware.

A clear key has not been encrypted under another key and has no additional protection within the cryptographic environment. For clear keys, the security of the keys is provided by operational procedures.

With the z10 GA3, the System z hardware adds support for protected keys. Protected keys blend the security of the Crypto Express3 (CEX3) and the performance characteristics of the CPACF. While a secure key is encrypted under a master key, a protected key is encrypted under a wrapping key that is uniquely created for each LPAR. The wrapping key is created each time an LPAR is activated or reset, and there are two variations of the wrapping key. One is for wrapping DES/TDES operational keys and a second for wrapping AES keys. The wrapping key is stored in the Hardware System Area (HSA) and is only accessible by firmware and cannot be read by the operating system or applications (even if running authorized).

If a CEX3 coprocessor is available, a protected key can begin life as a secure key. That is, ICSF will retrieve the secure key from the CKDS and the clear value will be recovered (decrypted from under the master key) and then that clear key is re-encrypted (or wrapped) under the appropriate wrapping key for that LPAR. The re-wrapping is managed by System z firmware in conjunction with the CEX3 coprocessor (CEX3C) and the use of a protected key is driven by the application when it invokes a CPACF API and specifies the label of a secure key. That wrapped key is passed back to ICSF which uses it within the CPACF to perform encryption/decryption operations. The underlying clear value of the key never exists in any address space, but only exists inside the secure hardware or the CPACF.

(There are new capabilities within RACF and the CSFKEYS profiles to restrict which secure keys can be used as protected keys. By default, all secure keys are considered SYMCPACFWRAP(NO), which means they are not eligible to be used as protected keys. The security administrator must define a CSFKEYS profile for the secure key and specify SYMCPACFWRAP(YES) in the ICSF segment before it can be used as a protected key.)

Alternatively, if no CEX3C is available, an application could use a clear key as the source for a protected key without using ICSF. In this case, the application is responsible for creating or loading a clear key value and then using the new PCKMO instruction (new on z10 GA3 - Nov. 2009 or later microcode) to wrap the key under the appropriate wrapping key.

Since the wrapping key is unique to each LPAR, a protected key cannot be shared with another LPAR. Either the original secure key or the clear key value, not the protected key, would be shared or stored. The underlying secure key or clear key would need to be retrieved and wrapped with the wrapping key within each LPAR.

A protected key can be a DES, TDES or AES key. It is important to realize that if the same underlying key value is used as a secure key, clear key or protected key, the resulting ciphertext will be identical. That is, the encryption operation will generate the

same results no matter which type of key is used. The difference is the level of protection provided by the hardware or operating system for the key values.

There are performance implications when using secure key, clear key or protected key. Beginning with the PCICC, the secure key APIs are implemented on the PCI cards, which require an asynchronous operation to move the data and keys from the general purpose CPs to the crypto cards, where the secure key work is performed. The clear key APIs are faster than the secure key APIs because the work is done at CPU speeds on the CPACF, which is a synchronous operation on the general purpose CP. Performance of protected keys will be closer to the speed of clear key. A protected key requires wrapping and unwrapping that is not done with clear keys, however it doesn't experience the delays associated with moving the data to be encrypted/decrypted to the PCI card. When a secure key is used as the source for a protected key, there will be some work done on the CEX3C to wrap that secure key under the wrapping key, however, once the key is wrapped, ICSF can keep the protected value in memory, passing it to the CPACF where the key will be unwrapped for each encryption/decryption operation.

It is also important to realize that the application design can have a significant impact on the performance characteristics. The crypto hardware (CPACF, Crypto Express2 and Crypto Express3) is designed to process large blocks of data. Passing large blocks of data to be encrypted or decrypted will provide noticeably better performance than passing small blocks, no matter where the work is done.

The choice of clear key, secure key or protected key is related to the security of the key, not necessarily the data. When you choose secure key, the tamper-sensing, tamper-responding hardware protects the keys. When you choose clear key, the hardware provides no specific protections for the key values, but your operational procedures provide that protection. (For examples: the data sets where clear keys are stored are SAF protected so that not just anyone has access to the key value; or dumps of the ICSF address space, which might contain a clear key value, are not accessible without the proper authority.) Protected keys provide a middle ground where the key is always encrypted under a wrapping key yet the keys can be used on the faster CPACF hardware. Depending on how you implement, there is still hardware protection while the key is stored as a secure key, and even when the key is decrypted from under the master key, the clear key value never exists in operating system or application storage.

The security requirements of the data must be the starting point for deciding between clear key and secure key. Factors such as the value of the data and potential impact of a compromise, the life-span of the data, the possible attacks that might be mounted against the data (insider attack vs. outsider attack) should also be considered when deciding how to implement encryption. The choice of secure key, clear key or protected key must be made by each organization on an application by application basis, considering the security requirements of the data, the performance requirements of the application and the operational procedures and security environment for the organization and the application. All of these could impact the decision on which hardware to implement.

Secure key is appropriate for applications that mandate that the cryptographic keys do not exist outside the secure boundary of the hardware (financial/ATM operations), i.e. those crypto operations which require the highest level of protection and can deal with the slower performance because of the operation of moving the request to the PCI card. Clear key is appropriate for applications that do not mandate the use of secure hardware to protect the key and require the highest level of performance. For example, data that is transient i.e. only exists briefly in storage or in transmission, may be able to take advantage of the better performance of clear key. Protected keys provide secure protection for the operational keys while at rest (i.e. stored in a data set) but use the CPACF to provide the best performance, so they may provide an acceptable alternative for secure keys for applications with strict performance requirements that might otherwise use clear keys.

In summary, secure keys provide no additional protection of encrypted data over clear keys. The encryption algorithms are the same for clear key and secure key. Secure keys do provide additional protection for key values within the operating system, but at the cost of performance. Each installation must weigh the cost of implementing policies and procedures for protecting all keys, especially clear keys, against the system resources required by secure keys.

For more details on secure key versus clear key, and the different crypto hardware devices, see Ernie's Nachtigall's TechDoc, TD101704 'Another Paper on CPACF Clear Key and CCF Functionality' for a look at the difference between CPACF and CCF in terms of Clear Key/Secure Key support. Also, reference WP100810, 'A Synopsis of System z Crypto Hardware'.