



A Brief Description Of The CKM[®] Process

The TecSec Constructive Key Management[®] (CKM[®]) product that incorporates a smart card is a software and hardware combination designed to cryptographically protect data from unauthorized access and use. It is a physical realization of Role Based Access Control concepts. When encrypting data, the user can selectively designate the read privileges (*Roles*) needed to read this data from amongst the 'Write' privileges that he possesses. The data is then accessible only by those authorized individuals who hold the correct 'Read' privileges; resulting in a one-to-many secure data distribution system. By being able to independently administer 'Read' and 'Write' access privileges, the abilities to create, modify and read data are treated as selectable privileges. Thoughtful use of Key Management processes in the construction of the keying material enables these security characteristics.

The heart of the system is the CKM[®] Combiner. Its function is to produce a working key for encryption, created through procedures incorporating encryption, random number generation, concatenation, and hashing. The inputs to this process are many but contain all of the credentials and permissions needed to access the target data. The permissions include a *Domain Value*, which is used to separate domains within the enterprise, and a *Maintenance Value*, which is used in the key update process. The working key is used to encrypt the target text and once accomplished, it is then destroyed.

The corresponding decryption key is re-created by the CKM[®] Re-Combiner using the corresponding set of 'Read' credentials and permissions, thereby revealing the data only to those with the correct set of access privileges¹. The decryption key is also destroyed following its one time use. The net result is that in the CKM[®] process data is secured by incorporating permissions into the key management such that the recipient has to have the correct credentials (to include 'Read', *Domain* and *Maintenance Values*) in order to recreate the decryption key and access the plain text.

The permissions, or privileges, for all participants are pre-established by a trusted third party, one of several CKM[®] system administrators. These administrators are created as part of setting up the overall Enterprise and are linked cryptographically such that impersonation would be extremely difficult. They are responsible for registering new *Members*, establishing *Member Credentials*, defining *Roles* and creating the necessary cryptographic supporting data. This information is then placed on *Tokens* in a secure manner, with the *Member* then using the *Token* to conduct his business. The participating *Members* of the Enterprise each have their rights and privileges (*Roles*) that are uniquely their own and protected by the full strength of the CKM[®] cryptography. The interactions between *Members* are limited to the intended interactions—the encryption and decryption of information by *Members* with overlapping *Roles*. There is an identity proof of membership required, partly based on possession of the *Token* and partly based on a Password, Biometric or an alternative Identification and Authentication process. Once the identity proofing has been completed, the Role-Based Access control defines a *Member's Role* (a cryptographic entity) within an organization or Enterprise, and the

¹ It is possible to have a set of permissions that does not allow you to 'read' (decrypt) the document you just wrote, though we suspect this will be an unusual situation with most deployments. This is completely analogous to the ordinary public key situation where Alice encrypts a message to Bob, and then she cannot read it unless she also encrypts it with her public key.



rights associated with that *Role*. The rules that control each *Member's* allowed actions, 'Read' (decryption) or 'Write' (encryption) or both, are also activated by successful *Member* identification. In particular this gives a *Member* his ability to read, create or modify documents in areas in which he has a *Role*.

SMART CARD "SILO" SECURITY

In the CKM[®] system, TecSec's hybrid enhanced Java Smart card is used in a multi-application configuration, Java Card System Standard 2.2² is assumed for this discussion. One smart card is able to contain many *Tokens* within its boundaries, thereby allowing the *Member*-owner to have many *Roles*, each isolated from the others by a combination of the CKM administration, the OS and hardware features of the card itself. A CKM Enabled[®] Silo then is a conceptualization of each independent *Token*, one of many, each with their own security protection. Applications could be as varied as allowing a *Member* with one card to have access to many compartments in a classified community, or, in a commercial environment, having several charge card images on the smart card or (stretching it a bit) a combination of the two.

While this CKM Enabled[®] Silo concept is unique to TecSec, all aspects of security are very important when it comes to the feasibility of a multi-application card. If multiple vendors each have their proprietary applications on a card, for example, they are going to want to know there will be no leakage of their secrets to a competitor. For security applications, it is essential that privileges allotted to one *Token* do not also apply to other segments of the card. CKM Enabled[®] Silos offer users both access control and confidentiality for each of the separate applications on the Smart card. For the Silo structure to be trusted, the security has to be implemented in a thoroughly high-assurance manner. All aspects of the system need to be considered, the CKM[®] administration and security as well as the overlapping security attributes of the card itself. CKM[®] administrative features that lend assurance to the Silo concept begin with the establishment of the Enterprise. The Enterprise, Domain and Organizational Units form the basic administration functions. All *Rights* (the ability to create another administrative element), *Roles* and security emanate from these basic functions. The creation of the Enterprise and its subordinate structure is accomplished under a rigid flow-down security scenario involving centralized management with audits monitoring actions of the participants. One of the critical parts of this is the granting of *Rights* to create new administrative elements. These can only be granted to a *Member* in good standing, thereby making him part of the administration and preserving the integrity of the management structure.

This process proceeds downward to the enrollment of the *Members* and assignment of *Roles* to each. The *Token* that is then issued represents the *Member's Permissions* or *Credentials* within any one *Role* that he may have. If the *Member* has multiple *Roles*, they have multiple *Tokens* within a single smartcard. The mechanics of the process result in their ability to encrypt and decrypt data according to the 'Read/Write' privileges established within their *Role*. All *Members* with the same *Roles* have the ability to access the same information. Changes to these *Roles* and/or *Tokens* are managed by the centralized Enterprise administration.

The significance of this process is that it provides centralized change administration but distributed control over the actions of each *Member* i.e., to 'Read' and/or 'Write' documents. This is instantiated in the Silo concept to provide total isolation between the Silo elements and high assurance that the system administration will not subvert the basic Silo security structure.

Specific security features that guarantee the integrity of the administrative process include:

² See Java Card Application Programming Interface



1. All administrators are identified and authenticated by certificates issued by an Enterprise Certificate Authority and a Public Key Infrastructure process protected by a secured management database.
2. Only *Members* in good standing can become new administrators.
3. Information about the *Rights* of each administrator is securely contained within the Enterprise management database. Also, if necessary, these *Rights* can be revoked within this database, thereby not requiring a *Token Update* or the permission of the administrator *Token* holder.
4. *Rights* to create new administrators are granted to each administrator only by his “supervisor,” providing an assurance thread back to the single, top Enterprise administrator.
5. In updating *Tokens*, identification and authentication of the administrator to the *Member Token* is accomplished through a Rivest, Shamir and Adleman (RSA) process as well as a hash that avoids substitution attacks and “phishing expeditions,” since errors will negate the current *Token* operation.
6. The downloading of new *Token* information to a *Member* is assured by using a CKM® process. This is essential to the secure isolation of the elements of the Silos.

The security features that guarantee isolation of a *Token* within its Silos from outside influences are:

1. *Token* access is restricted to those that can demonstrate an authenticated user *Role*.
2. Policies associated with *Credential*, *Domain* and *Maintenance* level expiration times are continuously checked with operations disabled in the event of violations.
3. Self Protection³ ensures that the *Token* is not modified by any means other than the processes provided by CKM® and the Smart card. This is mechanized by the TecSec Smart card Applet, the Java Card Operating System and the underlying Java Card hardware design. It involves the principles of Non-Bypassibility (no interaction between non-security-relevant interfaces and security functionality) and Non-Interference (Users are not able to modify the TecSec Smart card Applet or add additional applets to the Smart card).
4. The TecSec Smart card Applet performs authentication and stores current access rights on the *Token*. Its actions, in conjunction with the Java Card⁴, security provide the essential protection to the CKM® concept when applied to the multi-application smart card. The specific Applet functions that support both the card administration and CKM® combiner are:
 - 1) Key management, allowing generation of public/private RSA and Digital Signature Algorithm (DSA) key pairs, creating new key objects.
 - 2) Data signing, allowing for a means to sign data using RSA or DSA.
 - 3) Session key unwrapping, allowing for a means of decrypting single-part encrypted data using RSA.
 - 4) CKM® key derivation, allowing for a means of using pre-initiated *CKM® Domain* and *Credential* objects to derive a key.
 - 5) Key and data storage. The logical view of a *Token* is a device that stores objects and can perform cryptographic functions.
5. The TecSec Smart card Applet relies on functionality provided by the OS of the Smart card to:
 - 1) Protect the TecSec Smart card Applet itself from modification or deletion.
 - 2) Introduce application applets to the *Token*.

³ See CKM Run Time Environment

⁴ See Java Card Application Programming Interface



- 3) Check application applets for prior registration.
- 4) Limit the number of running applets to one (any applet by any vendor). This provides a single threaded system, thereby allowing no information bleeding between applets.
- 5) Isolate the memory space by assignment such that only one applet can access that memory.
- 6) Not allow Non-Volatile memory assigned to any applet to be reassigned.
- 7) Not allow any assigned memory to be read by card edge commands except through the controlling applet.
- 8) Ensure that most tampering events will be detected by the Hardware OS and will render the device non-operational.

This elaborate combination of tight security administration, combined hardware and software Silo isolation and fail-safe mechanisms will provide high levels of assurance, security and operational viability to the Silo concept as applied to multi-application smart cards.

FILE CLUSTERING CONSIDERATIONS

The CKM[®] system is a data protection mechanism in which information in the system is available to all Members at anytime but only those with the correct privileges are allowed access to their intended segments. Documents in their uniquely encrypted form can be available to all while pre-established *Roles* determine access to the plain text. These characteristics open up several applications for use that might not be as easily accommodated by more traditional systems. Among these are Analyst Data Sharing systems and Multi-Level Security (MLS) Data Distribution systems.

ANALYST DATA SHARING EMPLOYING CKM[®] TECHNOLOGY

In an Analyst Data Sharing system, there are multiple people involved, each with different tasks to perform. For example, there can be different collectors of information on a single subject, with each having their own area of expertise and each having *CKM[®] Tokens* defining their 'Write' privileges for a common document on this subject. A second group of *Members* would be analysts who could both 'Read' the multiple contributions and 'Write' their analyses and conclusions as to the meaning of the assembled information, i.e., generate the reports. Finally there are the policy makers or consumers of the reports who have *Read Credentials* only. Since the *Members* involved in this workflow will have many different privileges there may be occasional conflicts. For example, an analyst might not have all the *Credentials* needed to perform an analysis of a particular subject. Distribution of reports can also create problems in that all potential consumers may not have the *Credentials* needed for access to a final report. In the Homeland Security environment, for example, the flow of information from the collection at the front end of the intelligence world to the local law enforcement and first responder consumer are a prime example of this dilemma.

To resolve these problems, policy would need to be adjusted on a dynamic basis. The policy adjustments would have to be addressed by the information system managers, but the use of CKM[®] could ensure the security of the process. This information distribution would require other participants that we might call "Reviewers," -- *Members* that have the authority to sanitize and release selected information to analysts and consumers needing access but not holding the correct privileges. For these Reviewers to operate effectively, they may need the ability to grant temporary access under controlled conditions. This could involve a *Credential* request approval of a *Domain Authority* (DA) to issue a *Member* a *Credential* with limited rights to view only selected information, perhaps for only limited amounts of time. Both the policy adjustments and



the *Credential* issuance actions would have to be auditable by the *DA* or *Organizational Unit Authority* (OUA), a security imperative needed to maintain integrity in the system.

The above example shows that through the use of CKM®, complex MLS problems can be greatly simplified. CKM® would allow all *Members* involved to work with a common document, but each with their predefined *Roles*. Their *Tokens* would enable their *Roles*. The Silo concept would isolate *Tokens* physically and cryptographically such that sensitive compartments would not be revealed unless intended. Temporary policy adjustments would resolve the distribution and downgrading security problems.

For other systems without CKM® features to accomplish these same tasks, a nightmare of access control lists would be needed as well as a very complex process of selective encryption and distribution.

MLS DATA DISTRIBUTION SYSTEMS

In this case the object is to distribute one document to multiple *Members*, each holding different degrees of access privileges. It is similar to the Analyst Data Sharing application discussed above except that it would tend to be ‘read only’ and would involve selective downgrading. An example of this would be a Data Sharing Agreement between Coalition Partners. Different access policies would apply depending on the Political and Military situation. Again, policy makers would have to determine the rules applicable to each situation but CKM® could be employed to overlay and enforce the appropriate security posture.

Once again, each *Member* would only read the portions of the document that match his access *Role*, even though the total document contains all levels of information about the subject. The concept applied to this document then becomes a “System Low” document distribution process instead of “System High.” This has the implication that the document can be freely circulated without having to create specific circulation/access control lists with an assurance that only the intended recipients are able to view the appropriate information.

The access in this situation is predetermined by the centralized administrative structure prior to any operational phase, again acting under the concept of a centrally controlled management structure and distributed ability to grant access. The *Member* is able to select the audience for his document by selecting the *Credentials* to be used, thereby defining access rights by interest and not names. However, if the *Member* wishes to add or delete other *Members*, he would have to accomplish this through the *DA* and the *OUA*—the central managers. In the case of the Coalition Partners example above, groups could easily be added by adding another *Role* or *Credential* or even another *CKM® Domain*.

There are certain limitations here, however, because the *Token Update* process is in many ways a voluntary operation, it being one in which the target *Member* has to respond to an email for the change to become effective. This would allow a *Member* that wanted to maintain access to a certain *Role* to do so by ignoring a *Token Update* command, perhaps without the knowledge of the *Token Authority*. Access to all future data updates could be made unreadable through the use of the *Maintenance Level* update but that also requires cooperation by the target *Member*. Another way to solve this is to require short periods of credential validity, forcing all *Members* to update more frequently. For any downgrading required, special permissions would have to be given either to “write down” or “read up” by issuing *Token Updates* to the appropriate *Members*. This would be relatively easy to handle at the *Domain* level.

