International Journal of Engineering Applied Sciences and Technology, 2018
Vol. 3, Issue 2, ISSN No. 2455-2143, Pages 41-47
Published Online June 2018 in IJEAST (http://www.ijeast.com)

SECURITY OF PKCS#11 WITH HARDWARE SECURITY MODULES

Sahib Singh Juneja
Department of Information Technology
Amity University, Noida, India

Abstract-Cryptography is the boon of today's technology. All we need to do is save our data from other intruders. In this paper we will discuss the most important API standard which is PKCS#11, along with its integration with the hardware security modules. The main concern aligns in the fear of facing that the most secure PKCS#11 is also a victim of security attacks. We will discuss the attacks and find an appropriate solution to get prone of the attacks.

Keywords-cryptography; PKCS#11; security attacks; wrapping-unwrapping

I. INTRODUCTION

Every time we hear the word cryptography all that comes into mind is yes there is something that we do not want the world to see, there is something that we do not want any other person to use other than the one we wish for. So the question that comes is what basically is cryptography? What is that we actually do so that people do not get our information. Cryptography is nothing but the art of hiding our information, the art or science of modifying or information so that the unintended user cannot read it. Talking about the most live and basic example is suppose I want to have a chat with my best friend on WhatsApp and I want to tell her something very secretive .How do I know that a person sitting so far away from me is the intended person I am talking to?? The answer to this issue is when I met my friend a few days back we both decided to give each other a code. A code that only we both know. So now when we have to talk I just write suppose 010 (this is the code assigned to me) and from her end I would get suppose 015 (code of my friend). This is where I understand that yes I am talking to the intended person with whom I am supposed to share my talks with. This simple example leads to a very important concept of cryptography.

So basically cryptography is the study to transmitting data in a way that only the intended user can get the data. The most important question that comes is what made cryptography of prime importance in today's world? The need of 5 objectives made cryptography of prime importance. They being confidentiality, Integrity, Authentication, Non-Repudiation and the last being Authorization .The data to be sent must be highly confidential. No other user other than the intended receptor is supposed to get the data. This is of prime importance in today's world of secrecy Next that whatever data is received by the receiver the integrity of the data must be maintained. In other words the data must not be altered in any circumstances. When exchanging information a user must be able to identify himself to the receiver, that is he must me authenticated before any data is transmitted between them. In order to authenticate first of all the user must be authorized to be on the network or be in the ring of exchanging information. So cryptographic primitives fulfil the purpose of authorization as well. And the last thing is the Non-repudiation which says a sender cannot deny whatever he is sending. It is the concept of notarization that comes into picture.

Many algorithms were generated till date in order to get the encryption decryption done starting from the basic called ceaser cipher, playfair cipher, substitution ciphers, transposition ciphers, along with some advanced algorithms of DES, RSA, Deffie-Hellman, and many more. Data to be encrypted in small amounts is still a simple task but when we talk about enterprises that hold a large amount of data and that too sensitive data, the first and foremost requirement is of encrypting the data. Based of software techniques encryption is one of the way but then cryptographic events started taking place through hardware namely Hardware Security Modules (HSM's). The main reason of HSM's coming in picture is because as the amount of data increased, the length of keys used for encrypting increased, which after a particular time seems impossible to hold such large number of keys in a secure way too. To indulge the severity of securing large keys, HSM's proved to be a better option, reason being that is hardware.

II. WHY HARDWARE SECURITY MODULES

Routinely IT is looked with a choice on whether reason constructed appliances are desirable over programming. All things considered, reason manufactured appliances speak to another bit of

International Journal of Engineering Applied Sciences and Technology, 2018
Vol. 3, Issue 2, ISSN No. 2455-2143, Pages 41-47
Published Online June 2018 in IJEAST (http://www.ijeast.com)

physical equipment for the IT association to acquire, send, arrange, and keep up. More devices add to the capital consumption spending plan, add to the general IT many-sided quality (i.e., more bits of one of a kind equipment), and maybe even point of confinement arrangement adaptability inside the IT condition. With IT associations effectively battling with sizable and differing equipment inventories and possibly confined quarters, and quick to decrease their carbon impressions, a "more specific equipment" approach may not generally be the default decision.

In contrast to software, it has the benefit of introducing and running on conceivably existing and torpid servers and can ride the influx of enhancing server value execution and vitality efficiencies. Therefore, programming, at any rate at first from budgetary, IT operational, and carbon impression points of view, has all the earmarks of being a commendable other option to reason fabricated apparatuses. This careless perspective of equipment versus programming, in any case, has turned out to be less strong when the capacity being referred to is security. Most business and legislative elements perceive that security has one of kind properties that are hard to rope into the general IT condition while as yet keeping up useful uprightness. As confirmation of this, the market for reason constructed security machines is firmly positive. Where weight exists to reign in security machine sprawl, the bearings every now and again sought after are multi-useful security apparatuses (e.g., Unified Threat Management machines) or sharp edge and skeleton security stages. In the two occurrences, security capacities remain physically autonomous from whatever is left of the IT condition. HSM, as already portrayed, speaks to a vital component in ensuring digitized data. Endeavouring to achieve the same in programming ought not be managed without completely thinking about the suggestions. Following is our point of view on this issue.

III. The PKCS#11 Model

The model for PKCS#11 can be seen illustrated below, demonstrating how an application communicates its requests to a token via the PKCS#11 interface. The term slot represents a physical device interface. For example, a smart card reader would represent a slot and the smart card would represent the token. It is also possible that multiple slots may share the same token.

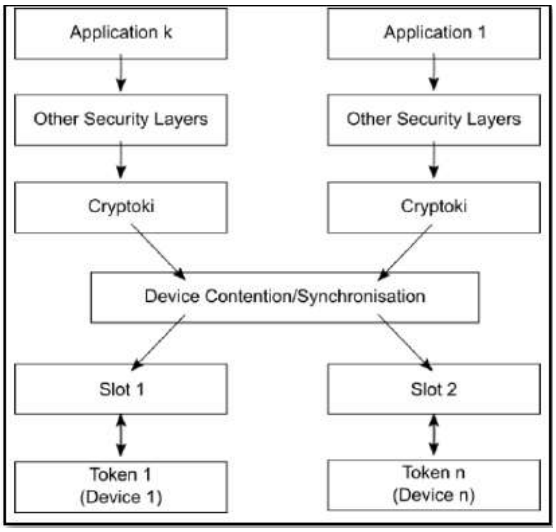


Figure 1: General PKCS#11 Model

Within PKCS#11, a token is viewed as a device that stores objects and can perform cryptographic functions. Objects are generally defined in one of four classes:

- Data objects, which are defined by an application
- Certificate objects, which are digital certificates such as X.509
- Key objects, which can be public, private or secret cryptographic keys
- Vendor-defined objects

Objects within PKCS#11 are further defined as either a token object or a session object. Token objects are visible by any application which has sufficient access permission and is connected to that token. An important attribute of a token object is that it remains on the token until a specific action is performed to remove it.

A connection between a token and an application is referred to as a session. Session objects are temporary and only remain in existence while the session is open. Session objects are only ever visible to the application that created them.

Access to objects within PKCS#11 is defined by the object type. Public objects are visible to any user or application, whereas private objects require that the user must be logged into that token in order to view them. PKCS#11 recognizes two types of users, namely a security officer (SO) or normal user. The security officer's only role is to initialize a token and set the normal user's access PIN.

IV. USE OF PKCS#11 IN HSM

SafeNet ProtectToolkit-C is a cryptographic service provider using the PKCS #11 application programming interface (API) standard, as specified by RSA Labs. It includes a lightweight, proprietary

International Journal of Engineering Applied Sciences and Technology, 2018
Vol. 3, Issue 2, ISSN No. 2455-2143, Pages 41-47
Published Online June 2018 in IJEAST (http://www.ijeast.com)

Java API to access these PKCS #11 functions from Java.
The PKCS #11 API, also known as Cryptoki, includes a suite of cryptographic services for encryption, decryption, signature generation, signature verification, and permanent key storage. The software found on the installation DVD is compliant with PKCS #11 v. 2.20. To provide the highest level of security, SafeNet ProtectToolkit-C interfaces with SafeNet access provider software and the SafeNet range of hardware security modules (HSMs):

- SafeNet ProtectServer Network HSM
- SafeNet ProtectServer PCIe HSM

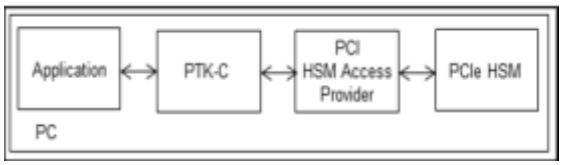
HSMs include high-speed DES and RSA hardware acceleration, as well as generic security processing. Secure, persistent, tamper-resistant CMOS key storage is included. Multiple adapters may be used in a single host computer to improve throughput or to provide redundancy. HSMs may be installed locally, on the same host system as SafeNet ProtectToolkit-C or they may be located remotely across a network.

Two product packages are available:

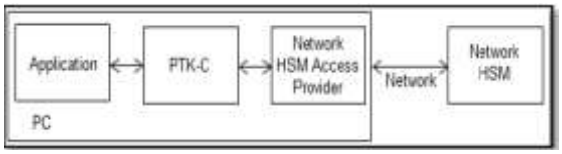
- Runtime for operational use
- Software Development Kit (SDK) for developer use

With SafeNet ProtectToolkit-C SDK installed, the API may operate in Software-Only mode for testing and development. In this mode, access to an HSM is not required.
The PKCS#11 Cryptographic Token Interface Standard, also known as Cryptoki, is one of the Public Key Cryptography Standards developed by RSA Security. PKCS#11 defines the interface between an application and a cryptographic device. This chapter gives a general outline of PKCS#11 and some of its basic concepts. If unfamiliar with PKCS#11, the reader is strongly advised to refer to *PKCS #11: Cryptographic Token Interface Standard*. PKCS#11 is used as a low-level interface to perform cryptographic operations without the need for the application to directly interface a device through its driver. PKCS#11 represents cryptographic devices using a common model referred to simply as a token. An application can therefore perform cryptographic operations on any device or token, using the same independent command set. SafeNet ProtectToolkit-C is a cryptographic service provider using the PKCS #11 application programming interface (API) standard, as specified by RSA Labs. It includes a lightweight, proprietary Java API to access these PKCS #11 functions from Java. The PKCS #11 API, also known as Cryptoki, includes a suite of cryptographic services for encryption, decryption, signature generation, signature verification, and permanent key storage.
HSMs include high-speed DES and RSA hardware acceleration, as well as generic security processing. Secure, persistent, tamper-resistant CMOS key storage is included. Multiple adapters may be used in a single host computer to improve throughput or to provide redundancy. HSMs may be installed locally, on the same host system as SafeNet ProtectToolkit-C or they may be located remotely across a network. SafeNet ProtectToolkit-C can be used in one of three *operating modes*. These are:

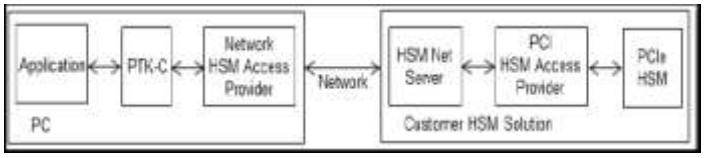
- **PCI mode** in conjunction with a locally-installed SafeNet cryptographic services adapter.



Network mode over a TCP/IP network, in conjunction with a compatible product such as the SafeNet ProtectServer PCIe HSM.



A machine with a SafeNet ProtectServer PCIe HSM installed may also be used as a server in network mode.



Software-only mode, on a local machine without access to a hardware security module.
Within the client/server runtime environment, the server performs cryptographic processing at the request of the client. The server itself will only operate in one of the hardware runtime modes.
The software-only version is available for a variety of platforms, including Windows NT and Solaris, and is typically used as a development and testing environment for applications that will eventually use the hardware variant of SafeNet ProtectToolkit-C.
Cryptoki Configuration
A number of steps must be taken in order for applications to operate correctly with SafeNet

International Journal of Engineering Applied Sciences and Technology, 2018
Vol. 3, Issue 2, ISSN No. 2455-2143, Pages 41-47
Published Online June 2018 in IJEAST (http://www.ijeast.com)

ProtectToolkit-C. TheSafeNet ProtectToolkit-C environment can be extensively configured in order to allow for the wide range of security requirements that various applications may have. It is important therefore that these requirements be known when configuring SafeNet ProtectToolkit so that the most suitable security settings and functionality for the particular applications can be chosen.
This chapter begins with an introduction to the application and security model used by SafeNet ProtectToolkit-C. The chapter then covers the steps required to configure a system utilizing SafeNet ProtectToolkit-C for the first time. The concepts of Trust Management and Token Replication are discussed and illustrated with examples

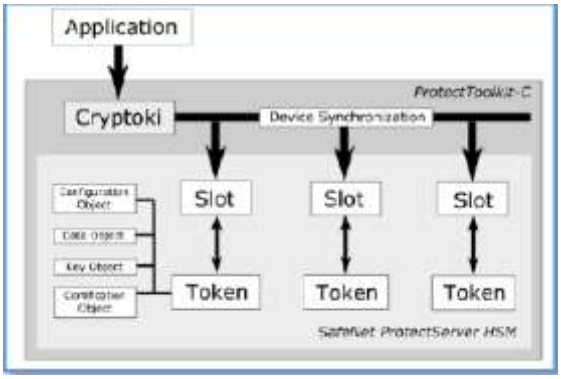


Figure-2 The safenet protect toolkit-c model

V. REASON OF INDULGENCE OF PKCS11

RSA Laboratories Public Key Standards (PKCS)#11 illustrates the 'Cryptoki' API, designed to be AN interface between applications and cryptological devices like smartcards, Hardware Security Modules (HSMs), and PCMCIA and USB key tokens. it's been wide adopted in business, promoting ability of devices. However, the API as outlined within the normal provides rise to variety of significant security vulnerabilities, [4]. In observe, vendors try and shield against these by proscribing the practicality of the interface, or by adding further options, the small print of that area unit typically onerous to see. This has crystal rectifier to AN unsatisfactory state of affairs in that wide deployed security solutions area unit mistreatment AN interface which is understood to be insecure if enforced naively, and that there aren't any well-established fixes. true is sophisticated by the range of situations during which PKCS#11 is employed as an efficient security patch for one situation might disable practicality that's very important for an additional.

VI. THE SECURITY OF PKCS #11

Security measures are as follows
A pin is always mandatory to have an access to private objects on token. Therefore in order to possess a cryptographic devices which implement a token will not be enough the pin is also an required attribute.
For enhanced protection private keys and secret key are marked not extractable as well as sensitive. The thing with sensitive keys is that they cannot be revealed in the plaintext of the token itself and similarly not extractable keys cannot be revealed even in encrypted form from the token.
These statements state that the main intention is that mark the objects as non -extractable as well as sensitive, and then any other user is not allowed to recover the secret values. This does not implies that our main motive is to prevent a user from using other users object those are private, so as it appears. The discussion between the designers concluded that there is main issue of concern which specifically include security of operating system, the threat that is posed by Trojan linked library, action that of rogue application, or the device drivers that may or may not subvert security, basically they do so by stealing of password .
Many reverent thoughts that relate to sniffing of communication line to cryptographic device do exist that in other words can be termed as eavesdropping. So here recover of PIN, or unauthorized access to any session where they can delete or modify or create any object and most importantly a token or device is impersonated all these issues are definitely compromised. But eventually the PKCS11 standard does claim that no attack that are discussed above can ever compromise key that are eventually marked as sensitive because in the end what sensitive means is a key will always no matter what will remain sensitive. Exactly the same way a key that is marked as non extractable will never be modified to be used as extractable. Therefore along with the examination of vulnerability of API we are definitely interested in the property that was claimed.
Now, any cryptographic devices which supports this standard which we are highlighting will face the leading threat model that are namely a fraud SO who basically abuse authorization of position of his as well as enabling of accessing user management functions and the device itself. Next can be a cheater user who basically will exploit his own authorizing access to token aand the last being a fraud second party which will gain access with one or other means to the tokens.
Most importantly, the threats we discussed above resolve by basically gaining access into session or to a device in between a session we can elaborate with an example such that by injection of message

International Journal of Engineering Applied Sciences and Technology, 2018
Vol. 3, Issue 2, ISSN No. 2455-2143, Pages 41-47
Published Online June 2018 in IJEAST (http://www.ijeast.com)

onto the communicating lines or simply knowing the password. It is very obvious that some very popular attacks are in general about speaking , as well as implementing dependencies as opposed to weakness in the APIs. Completeness was the basic definition for that.
The function of C_Login is highly vulnerable to an comprehensive PIN or in other words the password search since a user can try all possible passwords .One typical defence is to keep a count of the number of failed login attempts and 'lock' the card after a certain threshold of fails has been reached. Ideally, the counter should be incremented prior to testing the PIN and decrease only if successful. The attacker repeatedly and intentionally masquerades as the user an attempts to login with an incorrect PIN. An another way to do so is making buse of timedelays when the start up is done or in between login attempts.

```
Ck_define-function(ck_rv, c_login)
{
Ck_session_handle HSESSION
Ck_user_type USERTYPE
Ck_char_ptr PPIN,
Ck_ulong ULPINLEN
};
.
Ck_define-function(ck_rv, c_InitPin)
{
Ck_session_handle HSESSION
Ck_user_type USERTYPE
Ck_ulong ULPINLEN
};
```

Key Management Functions

- "C_GenerateKey"
- "C_GenerateKeyPair"
- "C_WrapKey"
- "C_UnwrapKey"
- "C_DeriveKey"

C_GenerateKey

```
C_GenerateKey(
CK_SESSION_HANDLE hSession
CK_MECHANISM_PTR pMechanism,
CK_ATTRIBUTE_PTR pTemplate,
CK_ULONG ulCount,
CK_OBJECT_HANDLE_PTR phKey
);
```

Description
This function operates as specified in PKCS#11. If the CKF_LOGIN_REQUIRED flag is set for the Token associated with the provided session the session state must be either CKS_RW_USER_FUNCTIONS or CKS_RO_USER_FUNCTIONS,otherwise the error CKR_USER_NOT_LOGGED_IN is returned.

C_GenerateKeyPair

```
CK_SESSION_HANDLE hSession,
CK_MECHANISM_PTR pMechanism,
CK_ATTRIBUTE_PTR pPublicKeyTemplate,
CK_ULONG ulPublicKeyAttributeCount,
CK_ATTRIBUTE_PTR pPrivateKeyTemplate,
CK_ULONG ulPrivateKeyAttributeCount,
CK_OBJECT_HANDLE_PTR phPublicKey,
CK_OBJECT_HANDLE_PTR phPrivateKey
);
```

Description
This function operates as specified in PKCS#11. If the CKF_LOGIN_REQUIRED flag is set for the Token associated with the provided session the session state must be either CKS_RW_USER_FUNCTIONS or CKS_RO_USER_FUNCTIONS,otherwise the error CKR_USER_NOT_LOGGED_IN is returned. If CKA_ID is not specified in either template then the library sets default values for these that are the same for both public and private object with a high likelihood of being unique. The value is a SHA1 hash of the modulus.

C_WrapKey

```
C_WrapKey(
CK_SESSION_HANDLE hSession,
CK_MECHANISM_PTR pMechanism,
CK_OBJECT_HANDLE hWrappingKey,
CK_OBJECT_HANDLE hKey,
CK_BYTE_PTR pWrappedKey,
CK_ULONG_PTR pulWrappedKeyLen
);
```

Description
This function operates as specified in PKCS#11. If the CKF_LOGIN_REQUIRED flag is set for the Token associated with the provided session the session state must be either CKS_RW_USER_FUNCTIONS or CKS_RO_USER_FUNCTIONS,otherwise the error CKR_USER_NOT_LOGGED_IN is returned.

C_UnwrapKey

```
C_UnwrapKey(
CK_SESSION_HANDLE hSession,
CK_MECHANISM_PTR pMechanism,
CK_OBJECT_HANDLE hUnwrappingKey,
CK_BYTE_PTR pWrappedKey,
CK_ULONG ulWrappedKeyLen,
CK_ATTRIBUTE_PTR pTemplate,
CK_ULONG ulAttributeCount,
CK_OBJECT_HANDLE_PTR phKey
);
```

Description

International Journal of Engineering Applied Sciences and Technology, 2018
Vol. 3, Issue 2, ISSN No. 2455-2143, Pages 41-47
Published Online June 2018 in IJEAST (http://www.ijeast.com)

This function operates as specified in PKCS#11.
If the CKF_LOGIN_REQUIRED flag is set for the Token associated with the provided session the session state must be either CKS_RW_USER_FUNCTIONS or CKS_RO_USER_FUNCTIONS,otherwise the error CKR_USER_NOT_LOGGED_IN is returned.

C_DeriveKey
C_DeriveKey(
CK_SESSION_HANDLE hSession,
CK_MECHANISM_PTR pMechanism,
CK_OBJECT_HANDLE hBaseKey,
CK_ATTRIBUTE_PTR pTemplate,
CK_ULONG ulAttributeCount,
CK_OBJECT_HANDLE_PTR phKey
);
Description
This function operates as specified in PKCS#11.
If the CKF_LOGIN_REQUIRED flag is set for the Token associated with the provided session the session state must be either CKS_RW_USER_FUNCTIONS or CKS_RO_USER_FUNCTIONS,otherwisthe error CKR_USER_NOT_LOGGED_IN is returned.
Simple derivation mechanisms are restricted to working on secret keys of type CKK_GENERIC_SECRET.
The functrion of C_WrapKey is basically used for conditions like if thewre is any need to wrap secret key with RSA public key or with any other secret key or the last option can be of wrapping an RSA, DSA or Deffie-hellman with secret key

VII. OPERATIONS ON PKCS#11

The first operation comes out to be of generating a key pair. In order to generate a key pair of private as well as public key the first condition is that public key must be sensitive and the attribute namely CKA_TOKEN must be set as true.
GEN_KEY_PAIR: H(NP; KP); PUB(KP); A(Np, T_PKCS),
SENSITIVE(NP, >T)

GEN_KEY is used to generate a symmetric key.

GEN_KEY: H(N1; K1); A(N1, T_SESSION)

The third operation is of wrapping a key which basically means to export a key and lastly of unwrapping a key which basically means importing a key

WRAP(ASYMMETRIC) : h(np; pub(kp)); h(n2; k2); wrap(np;>T),

exportable(n2;>T) –{K2}pub(kp)

WRAP(SYMMETRIC) : h(n1; k1); h(n2; k2); wrap(n1;>T),
exportable(n2;>T) –{|K2|K1}.

Attacks can be basically categorized into two streams:
Symmetric Key API Attacks
Public Key API Attacks
Symmetric Key API types are Key Conjuring, Key Binding, Key Separation, Weaker key or algorithm, Reduced key space, parallel search, Related Key attack.

Wrap Decrypt Attack
First we will discuss about the Wrap decrypt attack. Here the person intruding very well has the knowledge about the sensitive as well as extractable key in our case K1 and the wrap which is meant for decrypting key which in our case is K2. Now for a matter of fact we consider a key pair of RSA which is k2 , a public certificate that is included in the token will help the user to get the key and once he receives the key can generate a new RSA key pair Now the last key allows the intruder to export the key namely K1 and save it into a simple key. K1 is not actually in the encrypted due to the wrapping key exponential of one. Now a new key can be imported or created as validated by the token so that the exponent key will always remain a valid key.
The wrap/decrypt attack is as below:

EXPORT (ASYMMETRIC): H(N2; PUBLIC(K2)); H(N1; K1);(N1; EXPORTABLE); (N2; EXCHANGE_KEY) [{K1}PUB(K2)]

GET_EKEY [{K1}PUB(K2)] - {K1}PUB(K2)

ADECRYPT H(N2; PUB(K2));{K1}PUB(K2)

(N2, EXCHANGE_KEY) – K1

We must not use a key to be useful for both wrapping and decrypting because these add as attributes to set conflicts.

VIII. CONCLUSION

When considering the case of wrapped keys, with the aid of format change of external key token, we can easily deal with both Key Conjuring and Key Binding attacks. The introduction of proposals like [1] and [2] and under the tutelage and guidance from such prominent organizations like ANSI Financial Services Committee will aid in addressing the interoperability barriers. By executing a known implementation which prohibits the contrasting usage of key attributes (e.g. CKA_WRAP and CKA_DECRYPT), the Key Separation issue can be relatively dealt with. There is, however, a fundamental flaw in the design of wrapped key that it does not contain any separation information. Thus, the Key Conjuring and Key Binding attacks must be dealt with the help of a completely new external key token format. We can prevent the Weaker Key/Algorithm attack by concentrating on the fact that in no case should a key ever be protected/wrapped with the aid of a weaker algorithm/key. Additionally, protection against the Reduced Key Search attack can be achieved by accurately using the 'unextractable' and 'never extractable' flags. For future purposes, we will like to concentrate upon the matter that whether the mechanisms CKM_EXTRACT_KEY_FROM_KEY and CKM_XOR_BASE_AND_DATA (especially in its current format) should be considered or not for the API, as they could lead to a rise in Related Key and Parallel Search attacks. In order to prevent Private Key Modification attack, we can either take the aid of consistency check, thus ensuring key component integrity remains intact, or re-writing the format of the encrypted RSA key token which achieves integrity through cryptographic algorithms, like MAC over the token or encrypted hashing. We have seen the perils of using raw RSA functionality, which can lead to Small Public Exponent with No Padding attacks. The practical answer would be to impose the usage of a known padding configuration. The issue of backwards compatibility is yet to be dealt with. In this case, any device that uses this method to export a key would be left exposed to attack, so interoperability should not be a detriment. We need to set a standard for authenticating both public keys and wrapped keys (for import and export), in order to successfully deal with the threat of Trojan Public Key and Trojan Wrapped Key attacks.

IX. REFERENCES

[Jolyon Clulow. On the security of pkcs#11. In In proceedings of the 5th international workshop on cryptographic hardware and embedded systems (Ches'03), volume 2779 of lncs, pages 411{425.Springer-Verlag, 2003.

Riccardo Focardi, Flaminia L. Luccio, and Graham Steel. Foundations of security analysis and design vi. chapter An Introduction to Security API Analysis, pages 35{65. Springer-Verlag, Berlin,Heidelberg, 2011.

SafeNet inc. Attacking and _xing pkcs#11 security tokens a response By safenet. http://secgroup.ext.dsi.unive.it/ wpcontent/uploads/2010/10/Reponse-by-SafeNet.pdf.

RSA-laboratories. Pkcs#11: the cryptographic token interface standard. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/ v2-20/pkcs-11v2-20.pdf, 2004.

Graham Steel. Cryptographic key management apis. http://www.lsv.ens-cachan.fr/\~steel/teaching/pkcs11/ FMSS-crypto.pdf, 2013.