Bring Your Own Token (BYOT) to Replace the Traditional Smartcards

A case study on implementing BYOT solution with support for self-provisioning & management options to enable users to provision digital identities used for strong authentication and signing

Eric Hampshire
Cisco Systems

Karthik Ramasamy
Cisco Systems

Abstract

Smartcards are a good way to enable strong authentication to enterprise network and applications as they provide identification, authentication, and ability to store cryptographic key information on the card using the embedded microchip and memory. The enterprises can provision the smartcards with a digital identity, in the form of a X509 certificate uniquely associated with a user, to enable smartcard login to servers and Mutual TLS Authentication to services. Traditionally, hybrid cards that provides both the proximity card and smartcard functionalities are used for this purpose, so that the users can have a single card for both facility access as well as strong authentication to IT servers/applications.

There are some limitations and challenges with using the single card as both proximity and smartcard. The proximity cards can generally pre-provisioned in bulk as the association of the user identity to the proximity id can be done after the card is assigned to a user. But for the smartcard, the X509 certificates provisioned to the smartcards contain the user information that must be known at provisioning time. This slows down the provisioning process. There are also other challenges related to issuing replacement/temporary cards for lost or misplaced cards.

This whitepaper describes the solution implemented at Cisco, to replace the traditional hybrid smartcards with **Bring Your Own Token** (BYOT) model, to overcome the limitations and challenges with the traditional smartcard solutions. The solution enables users to bring their own USB tokens that are compatible with **Personal Identity Verification** (PIV) and **Chip Card Interface Device** (CCID) standards, to self-provision the digital identities needed to enable strong authentication, signing and other cryptographic functions.

Introduction

The idea of incorporating an integrated circuit chip onto a plastic card was first introduced by two German engineers in the late 1960s, Helmut Gröttrup and Jürgen Dethloff. Smartcards[1] are essentially a plastic card with an embedded integrated circuit chip. They come in many different types and dimensions, contact and contactless forms. Their physical characteristics, electrical interface, transmission protocols, crypto mechanisms, etc., are defined in the ISO/IEC 7810 and ISO/IEC 7816 standards. At their core they provide a tamper-resistant secure crypto processor and secure file system which communicates with external services using a card-reader. The cryptographic key material and digital certificates are securely generated/imported to the chip.

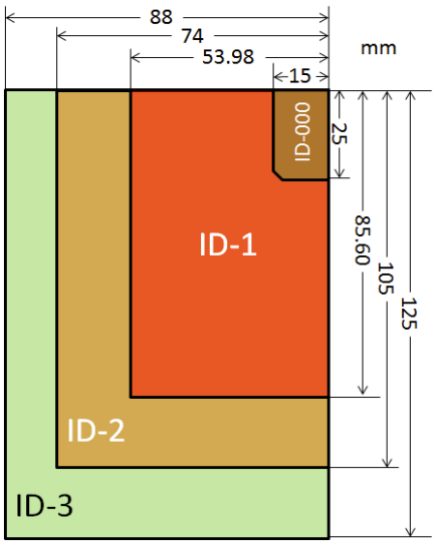


Figure 1: Smartcard Dimensions

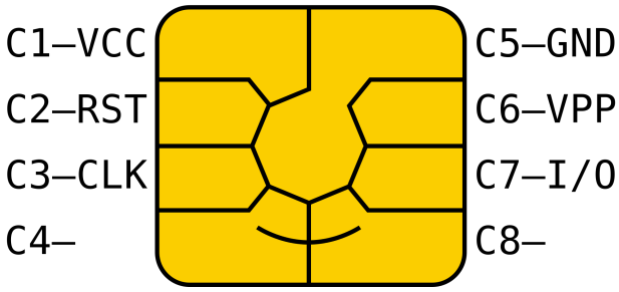


Figure 2: Smartcard pinout diagram

Smartcard Use Cases

One of the most widely adopted and successful use cases for smartcards is for mobile communications with the Global System for Mobile Communications (GSM)[2] network and the Subscriber Identity Module (SIM)[3]. Another use case is in banking and retail with EMV Chip cards[4] that implement the Chip & PIN or Chip & Signature schemes for securing credit transactions. Finally, we get to the focus of this whitepaper, the use case of smartcards for IT applications to enable strong authentication, signing, and encryption.

1 https://en.wikipedia.org/wiki/Smart_card

2 https://en.wikipedia.org/wiki/GSM

3 https://en.wikipedia.org/wiki/SIM_card

4 https://en.wikipedia.org/wiki/EMV

The traditional use of smartcards at large enterprises involved the use of hybrid cards that provide both the physical access proximity card and the logical smartcard functionality. This combined the two functions onto a single card for both the enterprise facility access and strong authentication to information systems. The digital identity certificates for strong authentication are either provisioned on premise in the badging office using kiosks or using 3rd party provisioning partners.

CCID and PIV Standards

The **Chip Card Interface Device** (CCID[5]) standard, defined by the USB standards work group in March 2001, specifies the protocol and requirements for a smartcard to be connected to a computer via a card reader using a standard USB interface, without the need for each smartcard manufacturer to provide its own reader or protocol. The latest revision of CCID (1.1) was released in April 2005.

The Personal Identity Verification (PIV) standard was published by National Institute of Standards and Technology (NIST) in February 2005 with **FIPS 201** document. This specifies the architecture and technical requirements for a common identification standard to improve the identification and authentication of Federal employees and contractors for access to Federal facilities and information systems. The latest revision of PIV (FIPS 201-2)[6] was published in August 2013.

The PIV standard specifies the requirements for standard slots/data objects in smartcards as listed in the table below. The commonly used slots are 9A and 9D.

Slot	Key Type	PIN Requirement
04	PIV Secure Messaging	Never
9A	**PIV Authentication**	**Once per session**
9B	PIV Card Application Administration	Never
9C	Digital Signature	Every use
9D	**Key Management**	**Once per session**
9E	Card Authentication	Never
82, 83, 84, 85, 86, 87, 88, 89, 8A, 8B, 8C, 8D, 8E, 8F, 90, 91, 92, 93, 94, 95	Retired Key Management Key	Once per session

Table 1: List of Standard Slots

These two standards have played a significate role in standardizing and simplifying the use of smartcards on computers for various use cases.

5 https://www.usb.org/sites/default/files/DWG_Smart-Card_CCID_Rev110.pdf
6 https://doi.org/10.6028/NIST.FIPS.201-2

Limitations with Traditional Smartcards

Most of the limitations with traditional smartcards revolve around the physicality of the smartcard. Programming the physical and logical portions of the card requires setting up kiosks in badging offices or using a 3rd party provisioning service provider. The 3rd party solution introduces a delay between requesting a badge and actual receiving a working, usable badge. It does, however, work pretty well for full-time remote workers. Conversely, a kiosk in a badging office on a company's campus does not work very well for full-time remote workers.

Another challenge is dealing with lost/misplaced smartcards. The same challenges in initial provisioning of a smartcard are in play here, with a user having to visit a badging office or wait for their new badge to be shipped. Tangentially related, if a strong identity for authentication is provisioned to the smartcard that the employee relies on for logical access to their workstation, forgetting the badge at home and arriving at the workplace can leave the employee without many good options. Most companies have some means to provision temporary badges for physical access, but what about the logical access?

Yet another challenge with the traditional smartcard program is the need for the card reader and the support for necessary driver/middleware on the host operating system to interact with the smartcard. The support for smartcard is not consistent across different operating systems. While the native support for smartcards is good on the Microsoft Windows, a middleware driver such as OpenSC[7] and/or additional vendor specific software is required to be installed on the MacOS and Linux to interact with smartcards.

Evolution of Strong Authentication at Cisco

Cisco has been using some form of strong authentication since 1997 and evolved over the years as show in the timeline chart below.

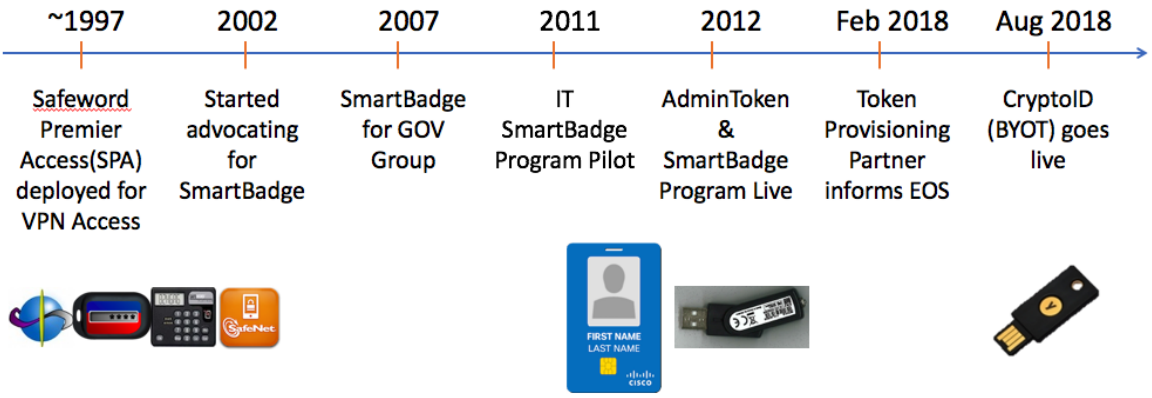


Figure 3: Timeline of Strong Authentication Solutions at Cisco

7 https://github.com/OpenSC/OpenSC

Safeword Premier Access

In 1997, Cisco deployed the Safeword Premier Access (SPA) product to enable users One Time Password (OTP) based authentication to protect sensitive and privileged access operations. This was used to protect the VPN Access to Cisco Networks, SSH access to remote servers and enabling root access (sudo).

The first couple iterations of this service offering were not self-service and required Cisco users to call into Cisco's internal helpdesk, Global Technical Resource Center (GTRC) for help with their tokens. SPA's OTP system was an Event-based OTP system (HOTP)[8] and therefore the counter between the clients and servers could get out of sync. Another problem was the counter for a particular user's token was required to be kept in sync between all the backend SPA servers, which was problematic when load was high and due to network latency.

GGSG Smartcard Program

In ~2003, the Infosec team at Cisco, who was responsible for the operation of the SPA servers at that time, began to evangelize a new combined physical and logical access token to replace the aging infrastructure and close some vulnerabilities. No executive was willing to pay the estimated ~$1m USD to replace all employee's badges, but a smaller group within Cisco, the Global Government Support Group (GGSG) was identified as a potential candidate for rollout of such a program in 2007. A Card Management System (CMS) was purchased from ActivIdentity[9] and ~300 users' badges were replaced with the new HID-sourced PIV-C smartcards.

Unfortunately, mistakes were made and the complexity, knowledge, and effort to maintain the CMS system was vastly underestimated. Proper resourcing was not assigned and ultimately the program failed with the configuration breaking and the limited resources being overwhelmed with user support requests.

Corporate Smartcard Program

In ~2012 approval was granted to stand up a Cisco corporate smartcard program that was labeled SmartBadge. The program relied on outsourcing the CMS and printing of the badges to a third party, IDonDemand (IDOD) which was later acquired by Identiv[10], as a managed service. Cisco still maintained and operated the PKI, requiring the use of an IDOD product called Bouncer to bridge the communication between IDOD and the Microsoft CA.

8 https://www.microcosm.com/blog/hotp-totp-what-is-the-difference

9 https://www.hidglobal.com/products/software/activid/activid-credential-management-system

10 https://www.identiv.com/services/

Due to the cost of the managed service, a portal was developed to handle enrollment of users and cross-charging of departments for the replacement badge. $85/year was the barrier for entry and, at the behest of the Cisco PKI team, was a prerequisite for also receiving an S/MIME certificate. This was meant to drive adoption of the SmartBadge as S/MIME certs had been requested by clients throughout the company for many years prior.

Ultimately, one thousand, eight hundred and fifteen (1,815) Cisco users participated in the badge replacement program at the $85/year cost. Many users did not finish the activation step of enrolling their new SmartBadge with the Physical Access System (PAC), Lenel, and simply spent the money to get a S/MIME cert so they could sign/encrypt their emails.

During this same period, the Yubikey NEO was evaluated as another option for a logical access token by the Cisco PKI team as a proof of concept. The PIV applet was provisioned with some test certs and authentication to various service was secured using them to prove out the concept.

AdminToken Program

A policy change made in the year 2015, by the Information Security group at Cisco, requiring a hardware token for Cisco administrative accounts on Microsoft Windows machines necessitated the development of a method to request and manage the new tokens. This was a natural second step following in the footsteps of the SmartBadge program. For this program, the SafeNet eToken was picked as the supported hardware token.

The provisioning and shipping of the tokens was still handled by the IDonDemand managed service with the same Cisco-run PKI on the backend. A separate portal was developed to manage the AdminToken requests, handling cross charges, approvals, and termination.

Need for a new solution

Several years after rolling out the SmartBadge and AdminToken programs, the adoption rate was still very small. This was mainly due to the expense and usability issues associated with both programs. In February 2018, IDonDemand informed Cisco that the managed service would be terminated in January 2019.

Thus, in early 2018, Cisco embarked on a new program, called CryptoID. The program, dubbed Bring Your Own Token (BYOT) was designed from the onset to enable Cisco employees, contractors or third parties to acquire their own token and provision a digital identity certificate. Cisco wanted this solution to be entirely self-service, avoiding the management overhead of cross-charging for hardware, and ultimately reducing support cases by empowering users to fix their own issues. The CryptoID program would replace the previously supported solution(s) into a single consolidated platform with support for different certificate types, such as UserToken, AdminToken and S/MIME.

Solution Description

The CryptoID solution is designed to be compatible with any USB Hardware Tokens that supports the **Personal Identity Verification** (PIV) and **Chip Card Interface Device** (CCID) standards. However, in order to ensure the **security and usability**, we will only onboard different token models after a thorough evaluation.

Functional Requirements

The following are the requirements for the proposed system.

- Ability to associate the token to a unique user
- Ability to control what token models, firmware versions are allowed to be registered
- Ability to control and manage the user and admin PIN policies such as retry count, length requirements, lock-out settings, etc.
- Ability to control the allowed cryptographic algorithms and mechanisms for the creation of key material and certificates
- Ability to ensure and attest that the cryptographic key materials are created on the hardware token and are not exportable
- Ability for users to create an authenticated and attested provisioning request and self-provision the token with the digital identities
- Ability for the users to be able to revoke or replace the certificates

High Level Design

The core components of this system are the CryptoID portal and the CryptoID client. These are the components that enables the BYOT and Self-provisioning aspects of this solution. These components interact with other existing IT systems such as Directory Services, Single Sign-On and Certificate Authority servers for the provisioning of the digital identities. The following diagram depicts the high-level system architecture of the CryptoID solution.

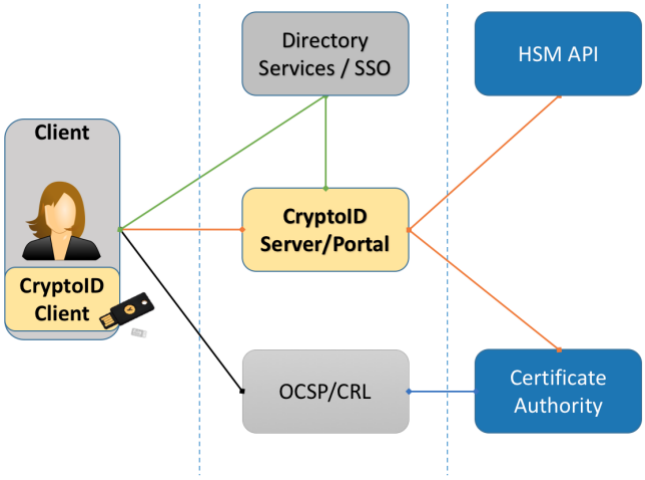


Figure 4: CryptoID System Design

Token Selection Criteria

One important aspect in the Bring Your Own Token approach is the decision process around which token vendors and models should be supported. The CryptoID solution is designed to be generic and extendible to work with any of the tokens that meet some of the basic requirements for the program. Cisco established a process and checklist to evaluate the hardware tokens against the core requirements for CryptoID program. Appendix F[11] provides an example evaluation form.

Based on our evaluation results, the Yubico YubiKey tokens proved to be reliable, low cost, and relatively easy to use. The YubiKey tokens meets all our core requirements for the PIV functionality and also provide other applets such as OTP and FIDO2. The OTP and FIDO2 can be configured with Cisco's Duo Single Sign On (SSO) system for second factor authentication. Cisco employees no longer have to chain together several different solutions to fit many use cases. Instead, they can entitle, enable and use ONE token for many security controls.

Advantages

The following are some of the advantages of the CryptoID solution.

- No external vendors involved in the provisioning process
- No vendor lock-in for the Hardware Tokens. The solution is generic and extensible.
- Flexibility to let users to bring their own tokens to provision the digital identities
- No need to revoke or replace the identities when the badge is misplaced or lost
- Quick turnaround thanks to self-provisioning
- Reduced cost and overhead

CryptoID Use Cases

The CryptoID solution currently supports provisioning of three different digital identities associated with a user. Support for additional identity types can easily be added.

- **AdminToken**
  - Issued to: Cisco Admins (CN=admin_userid@cisco.com)
  - Purpose: Administrative access to remote servers (smartcard login)
- **UserToken**
  - Issued to: Cisco Employees/Contractors (CN=userid@cisco.com)
  - Purpose: Strong Authentication (Smartcard Login or TLS Web Authentication)
- **SMIME**
  - Issued to: Cisco Employees/Contractors (CN=userid@cisco.com)
  - Purpose: Sending signed/encrypted emails

[11] Appendix F: Sample Token Selection Criteria Checklist Form

The users have complete control to request and manage their CryptoID certificates. They can request new certificates and manage their life-cycle; such as download, install and revocation. The CryptoID portal automatically publishes the issued certificates to Active Directory. It will also automatically revoke the certificates of any inactive Cisco employees.

Admin/User Token Use Cases

Both the AdminToken and UserToken identities are installed on the same slot (9a) in the token. So, the users who need both AdminToken and UserToken would require 2 different hardware tokens to work natively and comply with the PIV standards. The AdminToken and UserToken CryptoID certificates, installed on the token slot 9a, can be used for different strong authentication use cases such as:

- Smartcard Logon
  - The CryptoID certificate on slot 9a can be used to logon to Windows Servers
  - Windows natively supports smartcards complying with the PIV standards
- SSH Authentication
  - The CryptoID certificate on slot 9a can be used to strongly authenticate via SSH to a server running OpenSSH
  - Outside the scope of this whitepaper, there are many considerations and best practices for using the certificates and/or keys stored on a token in slot 9a[12]
- TLS Client Authentication
  - The CryptoID certificate on slot 9a can be used to strongly authenticate to a web server properly configured to accept certificates as an authentication option
  - Guides for configuring the TLS Client Certificate Authentication on the popular web servers such as Apache and Nginx web servers are readily available on the internet
  - For the client setup very little is needed on Windows/Mac OS with Chrome, IE, Edge, and Safari browsers, however Firefox requires some additional configurations on any platform as detailed in Appendix D[13]

S/MIME Use Case

The Secure/Multipurpose Internet Mail Extensions (S/MIME)[14] has been around since ~1997 and is widely supported in mail clients for sending signed and encrypted email. The CryptoID S/MIME certificate can be downloaded as PKCS#12[15] file that contains the private key, associated certificate, and certificate chain (Subordinate CA and Root CA certificates). It can either be imported to the token slot 9d or directly on the computer/mobile device's key store. This certificate can be used for signing and encrypting emails.

12 https://smallstep.com/blog/use-ssh-certificates/

13 Appendix D: Firefox Configuration for TLS Client Authentication

14 https://en.wikipedia.org/wiki/S/MIME

15 https://en.wikipedia.org/wiki/PKCS_12

Implementation Details

Overview

The CryptoID Portal and Client Tool were developed to give Cisco employees complete control in provisioning and managing their digital identities. The user simply procures any approved token (currently, YubiKey Series 4 or 5 with firmware v4.3.5 or greater), logs into the CryptoID Portal and follows the instructions to provision the required CryptoID certificate.

If a user needs both AdminToken and UserToken certificates, they would need to buy two tokens as both of those certificates are installed on the slot 9a to work natively and comply with the PIV standards.

AdminToken and UserToken Provisioning

The CryptoID Client provides a generic interface for the users to interact with the hardware tokens. It abstracts the token vendor specific middleware tools with the help of a standard client interface that is extensible to implement wrappers around the different supported token types. Currently it provides extensions for both yubico-piv-tool and ykman, provided by Yubico, to interact with the YubiKey tokens, through a simple command-line interface.

To provision a new YubiKey, the user downloads the client, runs the client tool, and chooses option #1 to generate a new CryptoID client request. The resulting Base64-encoded CryptoID Client Request[16] is displayed to the user for pasting back into the CryptoID Portal. The CryptoID design takes two steps to ensure the integrity and confidentiality of the request. We use the Attestation feature[17] of the YubiKey (firmware v4.3.5 or higher) to ensure the keypair that creates the Certificate Signing Request (CSR) was generated on the YubiKey. And, we authenticate the user to our Active Directory (AD).

The YubiKey PIV Attestation feature shows that a certain asymmetric key has been generated on device and not imported. This is accomplished by signing a specially-crafted Certificate Signing Request (CSR). The CSR is created from the newly-generated asymmetric key and includes specific extensions to capture the token details and then signed by the attestation certificate in slot f9. The attestation certificate comes pre-installed on the Yubikey and is issued by Yubico PIV CA. The CryptoID client includes this signature in the Base64-encoded CryptoID Client Request data.

As the CryptoID Client Tool is entirely separate from the CryptoID Portal and is run asynchronously, we authenticate the user to our Active Directory (AD) and include the username in the Base64-encoded CryptoID Client Request and on the attested Certificate Signing Request

16 Appendix B: CryptoID Provisioning Request Data

17 https://developers.yubico.com/PIV/Introduction/PIV_attestation.html

(CSR). The username in the request is verified against the username in the CSR and the user who logged in to the portal, to prevent any forgery.

Another precaution the CryptoID Portal takes is verifying the YubiKey token model and firmware through a whitelist. As the Attestation feature is only available on YubiKey firmware greater than 4.3.5 that is checked. Also, Yubico released a security advisory on 2017-10-16 about firmware versions lower than 4.3.5 producing weak RSA keys[18]. The PIN policy is also checked to ensure it is set to "Never:01", which disables the YubiKey requiring a PIN except on first use and allowing the application to cache it.

The provisioning process is broken into 3 high level tasks.

1. User downloads the CryptoID Client and runs the "**Generate CryptoID Provisioning Request"** Command**.** The client performs the following steps in the background to generate an attested CryptoID Provisioning Request.
    a. Verify YubiKey token connected
    b. Verify PUK requirements
    c. Verify Management Key requirements
    d. Verify PIN requirements
    e. Authenticate User to Active Directory (AD)
    f. Generate Keypair on Token
    g. Generate CSR from keypair on token
    h. Attest keypair on token
    i. Build Base64-encoded CryptoID Provisioning Request and display to user
2. User submits the attested CryptoID Provisioning Request data to CryptoID Portal. The portal performs the following steps to validate and issue the CryptoID.
    a. Verifies userID for logged-in user and userName in request match
    b. Verifies Certificate Signing Request (CSR)
    c. Verifies Attestation in request
    d. Submits CSR to Certificate Authority (CA) over REST
    e. Receives X509 Certificate from CA
    f. Provisions X509 Certificate to User's Active Directory (AD) account
3. User installs the X509 Certificate into the token's PIV authentication slot (9a)
    a. User downloads X509 Certificate
    b. User installs X509 Certificate using CryptoID Client

18 https://www.yubico.com/support/security-advisories/ysa-2017-01/

S/MIME Provisioning

The S/MIME certificates can be provisioned directly on the CryptoID Portal. The process does not require any request data to be generated using the CryptoID client. Cisco escrows the private key associated with the user's S/MIME certificates for legal purposes. So, the private key and CSR for the S/MIME certificate is generated securely on the CryptoID server using a Hardware Security Module (HSM) and its underlying True Random Number Generator (TRNG)[19]. The CSR is then submitted to the CA to issue the cert and finally a P12 file is created. The P12 is protected using a secure random password. Both P12 and the P12 password are encrypted using an AES-256 key before storing in the database. The AES-256 key used for the encryption is securely generated on an HSM and marked non-exportable, non-extractable.

The user can download the P12 file, and the associated password from the CryptoID portal, and import it either to their computer/mobile key store, or to their YubiKey slot 9d.

All the S/MIME certificates provided through the CryptoID portal have a 2-year validity period. This short validity period was chosen because the certificates are provided in a portable format (P12) purely in software. The user is emailed through an automated process to be notified their certificate is expiring in 90 days, 60 days, 30 days, 7 days, and 1 day.

De-Provisioning

Users have control to revoke their own certificates and there is a background task that revokes certificates for inactive Cisco employees, querying Cisco's Active Directory (AD) nightly. If a user loses their token, we are relying on them to notice, return to the CryptoID Portal and revoke the credential. They must revoke the current credential in order to procure a new one - i.e. users cannot have more than one active AdminToken, UserToken, or S/MIME.

One Time Password (OTP) Seed Provisioning

YubiKeys come with a One Time Password (OTP) applet[20] and the seed pre-provisioned to the YubiCloud[21] for authentication. Cisco uses Duo[22] for our Single Sign On (SSO) infrastructure. Duo supports YubiKey OTP as second factor authentication mechanism but it does not use the YubiCloud as a backend authentication service. Instead it requires the YubiKey OTP tokens to be individually registered with the OTP seed. Duo provides an API for enabling this provisioning.

19 https://en.wikipedia.org/wiki/Hardware_random_number_generator

20 https://developers.yubico.com/OTP/OTPs_Explained.html

21 https://www.yubico.com/products/services-software/yubicloud/

22 https://duo.com/

The CryptoID Client Tool was enhanced to generate this OTP seed and provide a Base64-encoded request[23] as output to be copied by the user into the self-service Duo registration portal[24].

Similar to the AdminToken and UserToken provisioning process, the user initializing their OTP seed is authenticated to Cisco's Active Directory (AD) and their username is included in the Base64-encoded request data. As the Yubico tools provided no way to protect the OTP seed, an additional step to encrypt the seed value to the CryptoID Portal's RSA Public key was taken as a security measure.

Libraries, Tools, IDEs

The CryptoID Portal was developed entirely using Java, targeting the Java v8 JRE. The team makes extensive use of the IAIK JCA/JCE Toolkit[25] as a crypto utility. The IAIK Toolkit is very similar to the more popular, open-source BouncyCastle[26], just with better documentation. The IDE used was the excellent Eclipse Foundation[27] or a variation of.
The CryptoID Client bundled the yubico-piv-tool[28] compiled for Windows, Linux, and MacOS target client platforms. See Appendix A for detailed commands used.

Development Time

Two people from the Cisco PKI team developed CryptoID over 4 months to get to the v1.0 release. A test version was ready in ~2 months for a limited release scope (10-15 users) and much positive feedback was received. This timeline was greatly compressed because of the various lessons learned from the previous years with the various strong authentication programs and the team's familiarity with operating a PKI. CryptoID is now at v1.4 and has had very little in terms of upkeep and maintenance for the developers. Future enhancements have been thought through and are covered in the next section.

23 Appendix C: OTP Registration Request Data

24 https://disco.cisco.com/

25 https://jce.iaik.tugraz.at/sic/Products/Core_Crypto_Toolkits/JCA_JCE

26 https://www.bouncycastle.org/

27 https://www.eclipse.org/

28 https://developers.yubico.com/yubico-piv-tool/

Future Enhancements

Other Token Types

Hardware security keys like those from Feitian[29], PIVKey[30], and SafeNet eToken[31] would be nice to support in addition to YubiKeys, giving Cisco users many form factor choices for what they would like to use. The primary roadblock to supporting more token types is the development and support of the CryptoID Client. Also, the attestation feature is something unique to the YubiKeys Cisco would miss out on implementing support for other token types. This limitation could be partially overcome by the CryptoID Client implementing a similar feature in software.

Platform Native CryptoID Tools

For the One Time Password (OTP) seed provisioning, a standalone utility was developed in Python to utilize the ykman Python libraries[32] and then compiled to target the client platforms using pyinstaller[33]. This stand-alone utility was then signed with the platform appropriate code sign utility for distribution. A future enhancement to the CryptoID Client Tool would be to re-write it in Python and compile a platform-specific version as well.

Alternatively, a GUI version developed using Electron[34] would be nice for those employees not comfortable with command-line utilities. Advantages of the thick client or platform-specific binaries would include the elimination of a copy/paste step from the Client to the Portal, the clients could speak directly to the Portal via APIs.

S/MIME Certificates for Vanity Email Aliases

With the development of CryptoID, Cisco added the option of a free S/MIME cert to any Cisco employee using their username@cisco.com email address in both the Email (E) relative distinguished name[35] field of the SubjectDN[36] and the rfc822 subject alternative name[37] field. Almost immediately, we had requests to support vanity email aliases that the Cisco users had procured over the years. Supporting these vanity email aliases would be a good feature to include in the future.

29 https://www.ftsafe.com/Products

30 https://pivkey.com/

31 https://safenet.gemalto.com/multi-factor-authentication/authenticators/pki-usb-authentication/

32 https://github.com/Yubico/YubiKey-manager/tree/master/ykman

33 https://www.pyinstaller.org/

34 https://electronjs.org/

35 https://ldapwiki.com/wiki/Relative%20Distinguished%20Name

36 https://ldapwiki.com/wiki/Distinguished%20Names

37 http://www.pkiglobe.org/subject_alt_name.html

S/MIME Certificates for Mailers

There are many email messages sent to Cisco employees from mailing lists so that replies to the email go to a group of people. We have received a few requests for these mailers to be assigned S/MIME certs so the emails can be signed and look even more "official". So far, we have been able to support these requests in a very manual way by going directly to the CA to issue the certs but would like to enhance the CryptoID Portal to allow owners of mailers to "enroll" on behalf of their various mailers. This would let us keep track of these certificates, owners could get expiry notifications, and certs would be revoked if mailers were deleted or owners left Cisco.

UserToken for Generic Accounts

Active Directory (AD) generic accounts are generally used by automated scripts/processes to authenticate with remote resources. It would be nice to strongly authenticate these accounts with an X509 certificate much like we can strongly authenticate regular Cisco users. Cisco has a web application for owners to easily manage these accounts owners, members, and passwords. The CryptoID Portal could tie into this API to ensure UserTokens were being requested only by generic account owners. Also, this would let us keep track of these certificates, owners could get expiry notifications, and certs would be revoked if generic accounts were deleted or owners left Cisco.

Lessons Learned / Key Considerations

Token use in VMware

Requiring Windows administrators to login with their PIV-enabled YubiKey quickly uncovered the fact that many of Cisco's Windows admins are using a different OS than Windows on their workstations (MacOS, Linux, etc.). When attaching the YubiKey to a VM in VMware the default behavior is for that USB device to show as "shared". Yubico had a solution involving modification of the VMware VMX file for the VM[38], marking the YubiKey as dedicated to the VM and allowing Windows to access the PIV authentication slot (9a) on that YubiKey for authentication to the user's privileged administrator account.

MacOS Catalina Issues

A very recent problem has been Apple's release of MacOS Catalina and the disabling of 32-bit applications. Both the CryptoID Client Tool and the standalone platform-specific compiled binary for the OTP seed provisioning have encountered problems with various client installs of the new MacOS. More investigation is needed and possibly new releases of these tools.

38 Appendix E: VMware Configuration for dedicated YubiKey

Involvement of Infrastructure Teams

When considering standing up a similar program at your company to offer hardware-based, strong identity certificates, it is wise to involve your Active Directory (AD) and Public Key Infrastructure (PKI) technical points of contact early. Cisco was lucky in that the developers on CryptoID were the PKI guys, so integrations was easy. Having a good working relationship with the AD guys certainly helped the integrations there, along with a generic account that had elevated privileges to publish certificates to all user and administrator accounts in AD.

Return on Investment (ROI)

Token Reliability

The YubiKeys have proven to be much more resilient and harder to break. Designed to be added to keyrings, they are built for quite a bit of abuse and we are not getting very many reports of damaged tokens. We have only received reports of couple of faulty YubiKey tokens in the last year, compared to more than 50 token replacement requests per year with the old AdminToken program.

Token Consolidation

The multi-protocol support provided by YubiKey is very helpful to enable the same hardware token to be used for different use cases. Apart from the PIV protocol that's used for the CryptoID solution, YubiKey supports OTP and FIDO2 protocols, which is integrated with Cisco's Single Sign On (SSO) service as an optional second factor authenticator.

Licensing/Maintenance Costs

Setting up the dedicated environment (CMS, hardware, etc.) for Cisco cost ~$250k, which covered the first year of maintenance. On top of that Cisco paid ~$20/user/year for the "subscription" to the managed service. That worked out to be ~$50k/year just for the SmartBadge program.

The old managed service for AdminTokens was quite a bit more expensive, costing Cisco ~$300k/year in licenses and maintenance. All of this and the SmartBadge cost were avoided by moving to our in-house, home-grown solution.

Support Cases

The comparison of number of support cases for the old AdminToken and SmartBadge system versus the new CryptoID solution with YubiKeys is shown in Table 2: Comparison of Support Case Metrics. While difference in number of cases is not very significant, the number of cases reduced significantly over time, especially in the recent months, as shown in *Figure 5: Breakdown of CryptoID cases by Cisco Fiscal Quarter*.

The CryptoID has been more reliable and easier to adapt for the users and it saved Cisco some dollars/time.

	Cases/Quarter
Admin Token + Smart Badge	80-85
CryptoID	60-65

Table 2: Comparison of Support Case Metrics

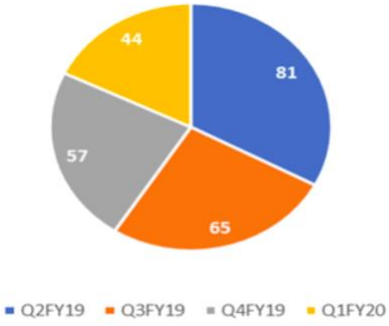


Figure 5: Breakdown of CryptoID cases by Cisco Fiscal Quarter

Conclusion

In this whitepaper we have presented the CryptoID solution that Cisco implemented to replace the traditional smartcards with the Bring Your Own Token support. The rollout of the CryptoID program has resulted in reduced overhead and increased efficiency to manage the tokens and the digital identities. The best practices and lessons learned from this implementation can help other organizations that plan to implement a similar solution.

Appendix A: yubico-piv-tool and ykman Commands

The CryptoID Client provides a menu-driven interface for the users to interact with the tokens. The CryptoID CLI menu is shown in the image below. The initial version of CryptoID Client used the yubico-piv-tool to interact with the YubiKey tokens. Since Yubico decided to consolidate their client development on the newer ykman[39] tool, a version of the CryptoID Client has been developed to add support for ykman.

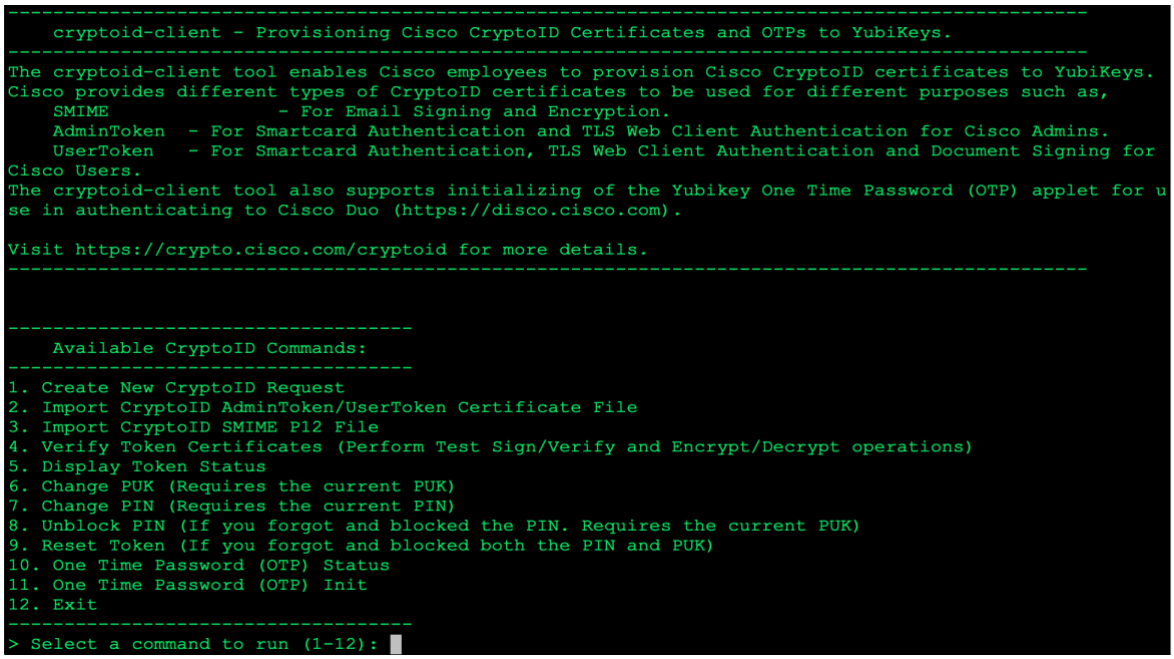


Figure 6: CryptoID Client – Menu List

This section provides detailed steps and the **yubico-piv-tool** / **ykman** commands implemented behind the scenes for each of the menu options in the CryptoID client.

1. Create New CryptoID Request

1.1. Verify Token Connection

yubico-piv-tool -a list-readers
ykman list

1.2. Verify PUK Requirements

NOTE: The yubico-piv-tool doesn't provide a verify PUK function, so as a workaround, we use the change-puk method to verify

yubico-piv-tool -a change-puk -P <PUK> -N <PUK>
ykman piv change-puk -p <PUK> -n <PUK>

39 https://github.com/Yubico/YubiKey-manager

1.3. Verify Management Key Requirements

NOTE: The yubico-piv-tool doesn't provide a verify PUK function, so as a workaround, we use the change-puk method to verify

yubico-piv-tool -a set-mgm-key -k <Mgmt_Key> -n < Mgmt_Key>

ykman piv change-management-key -m < Mgmt_Key> -n < Mgmt_Key>

1.4. Verify PIN Requirements

yubico-piv-tool -a verify-pin -P <PIN>

ykman piv change-pin -P <PIN> -n <PIN>

1.5. Generate Keypair

yubico-piv-tool -a generate -k <Mgmt_Key> -s 9a --pin-policy=once --touch_policy=never -A RSA2048 -o <publicKey_output_file>

ykman piv generate-key --pin-policy=once --touch-policy=never -a RSA2048 9a <publicKey_output_file>

1.6. Generate CSR

yubico-piv-tool -a verify -a request-certificate -k <Mgmt_Key> -P <PIN> -s 9a --hash SHA512 -I <publicKey_output_file> -o <csr_output_file> -S <subjectDN>

ykman piv generate-csr -P <PIN> -S <subjectDN> 9a <publicKey_output_file> <csr_output_file>

1.7. (Attest Request)

yubico-piv-tool -a attest -k <Mgmt_Key> -s 9a -o <keyAttestion_output_file>

ykman piv attest 9a <keyAttestion_output_file>

2. Import CryptoID AdminToken/UserToken Certificate File

2.1. Import Certificate File

yubico-piv-tool -a import-certificate -k <Mgmt_Key> -s 9a -i <file_location>

ykman piv import-certificate -v -m <Mgmt_Key> 9a <file_location>

2.2. Change CHUID

yubico-piv-tool -a set-chuid -k <Mgmt_Key>

ykman piv set-chuid -m <Mgmt_Key>

3. Import CryptoID SMIME P12 File

3.1. Import SMIME Certificate File

yubico-piv-tool -a import-key -a import-cert -k <Mgmt_Key> -s 9d -i <p12_file_location> -K PKCS12 -p <p12_password>

ykman piv import-certificate -v -m <Mgmt_Key> -p <p12_password> 9d <p12_file_location>

3.2. Change CHUID)

yubico-piv-tool -a set-chuid -k <Mgmt_Key>
ykman piv set-chuid -m <Mgmt_Key>

4. Verify Token Certificates (Perform Test Sign/Verify and Encrypt/Decrypt operations)

4.1. Test Sign and Verify

NOTE: The ykman tool doesn't provide a verify and sign function
yubico-piv-tool -a verify-pin -a test-signature -s 9a -i <slotCert_output_file> -P <PIN>

4.2. Test Encrypt and Decrypt

NOTE: The ykman tool doesn't provide an encrypt and decrypt function
yubico-piv-tool -a verify-pin -a test-decipher -s 9a -i <slotCert_output_file> -P <PIN>

5. Display Token Status

yubico-piv-tool -a status
ykman piv info

6. Change PUK (Requires the current PUK)

yubico-piv-tool -a change-puk -P <current_PUK> -N <new_PUK>
ykman piv change-puk -p <current_PUK> -n <new_PUK>

7. Change PIN (Requires the current PIN)

yubico-piv-tool -a change-pin -P <current_PIN> -N <new_PIN>
ykman piv change-pin -P <current_PIN> -n <new_PIN>

8. Unblock PIN (If you forgot and blocked the PIN. Requires the current PUK)

yubico-piv-tool -a unblock-pin -P <current_PUK> -N <new_PIN>
ykman piv unblock-pin -P <current_PUK> -n <new_PIN>

9. Reset Token (If you forgot and blocked both the PIN and PUK)

yubico-piv-tool -a reset
ykman piv reset -f

10. One Time Password (OTP) Status

NOTE: The yubico-piv-tool doesn't provide OTP functions
ykman otp info

11. One Time Password (OTP) Init

NOTE: The yubico-piv-tool doesn't provide OTP functions
ykman otp yubitop -G -g -S -f <slot>

Appendix B: CryptoID Provisioning Request Data

The CryptoID Provisioning Request Data is organized as a simple JSON structure with the following elements:

- **userName**:
  - Cisco userID that was authenticated against AD to create the request
- **tokenType**:
  - Token Type
- **csr**:
  - Certificate Signing Request generated from keypair on token
- **attestionData**:
  - output from "attest" command given to YubiKey
- **attestionSigner**:
  - read from slot f9 of the YubiKey
- **clientHeuristics**:
  - userName, hostName, and hostAddress of the machine used to run the client

Given below is a sample CryptoID Provisioning Request Data.


```
{
 "username": " userid",
 "tokenType": " Yubico YubiKey",
 "csr":"-----BEGIN NEW CERTIFICATE REQUEST-----
                <csr data>
        -----END NEW CERTIFICATE REQUEST-----",
"attestationData":"-----BEGIN CERTIFICATE-----
                <attestation data>
         -----END CERTIFICATE-----",
"attestationSigner":" ---BEGIN CERTIFICATE-----
                <attestation signer data>
         -----END CERTIFICATE-----",
"clientHueristics": {"hostName":"host1", "hostAddress":"192.168.1.193", "userName":"userid"}
}
```

Appendix C: OTP Registration Request Data

The OTP Registration Request Data is organized as a simple JSON structure with the following elements:

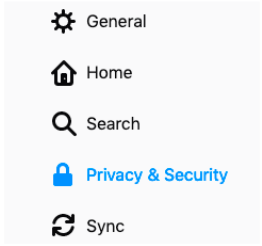
- **userName**:
  - Cisco userID that was authenticated against AD to create the request
- **serialNumber**:
  - YubiKey serial number
- **firmwareVersion**:
  - YubiKey firmware version
- **deviceType**:
  - Token Type
- **publicId**:
  - First 12 characters of the string output by a YubiKey when pressing it
- **privateId**:
  - Can be accessed when an OTP is decrypted in a Yubico OTP validation server
- **secretKey**:
  - The actual OTP seed value

The entire structure is then encrypted using RSA/ECB/OAEPWithSHA1AndMGF1Padding to an RSA public key held by the registration portal. Given below is a sample OTP Registration. Request Data.

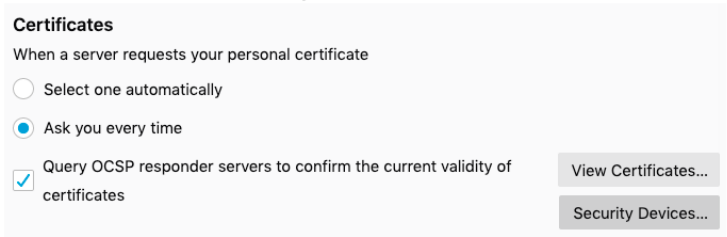

```
{
  "deviceType": "YubiKey 4",
  "firmwareVersion": "4.3.7",
  "privateId": "9e14c1c2850e",
  "publicId": "fetchchnkglk",
  "secretKey": "3eb089934ee7c27091cb89b39942e1a9",
  "serialNumber": "7050665",
  "userName": "userid"
}
```

Appendix D: Firefox Configuration for TLS Client Authentication

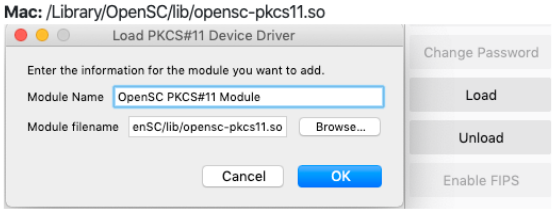
1. Install OpenSC for your platform:
   - Mac - OpenSC
   - Windows - OpenSC
2. Firefox has it's own store for certificates, so you need to tell it to look at your Yubikey as a "source". Go into Firefox Options/Preferences and the Privacy and Security tab:



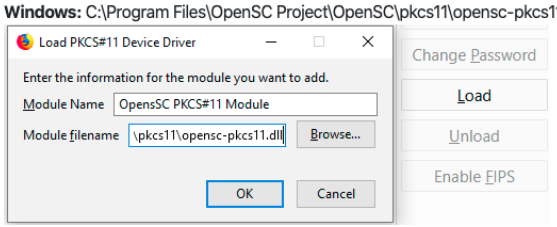
3. Scroll to the bottom and click on "Security Devices..." under the Certificates section:



4. Click "Load" (on the left) and add the following path appropriate to your platform:
   - **Mac:** /Library/OpenSC/lib/opensc-pkcs11.so



   - **Windows:** C:\Program Files\OpenSC Project\OpenSC\pkcs11\opensc-pkcs11.dll



Appendix E: VMware Configuration for dedicated YubiKey

In order for the PIV protocol functions to work correctly with the YubiKey tokens on a Virtual Machine running on VMWare Fusion or VMWare Workstation, the YubiKey must be connected to the guest VM in dedicated mode. If it gets connected on “Shared” mode, the VM will not detect the PIV slots. On some versions of VMware Fusion and VMware workstation, it is required to make some configuration changes to enable connecting the YubiKey in dedicated mode.

To verify if you need these changes or not, check if you are getting an option to connect the YubiKey in dedicated mode to the VM as shown in the picture below (highlighted menu option). If you only see the option to connect in “Shared” mode, then try following the instructions in this section.

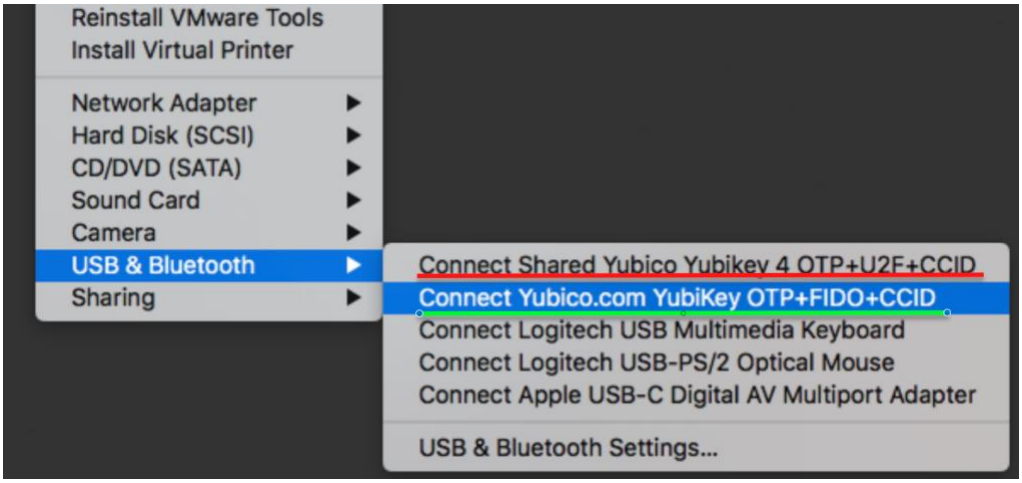


Figure 7: Connecting YubiKey in "dedicated" mode

Follow these steps to make configuration changes needed to connect the YubiKey in dedicated mode to a Virtual Machine running on VMware Fusion or VMware Workstation

1. Ensure the VMware Tools are installed on the VM and power down the VM
2. Check the Virtual Machine Details section in VMware Workstation / VMware Fusion and note where the Configuration file (.vmx) is located, as shown in the image below.

Figure 8: Location of the VM Configuration File

3. Open the VM configuration [virtualmachinename].vmx file in a text editor.
4. Paste the below lines at the very end of this file and save the file.

   machine. usb.generic.autoconnect = "FALSE"
   usb.generic.allowHID = "TRUE"
   usb.generic.allowLastHID = "TRUE"
   usb.generic.allowCCID = "TRUE"

   ***Note**: The first line (usb.generic.autoconnect) is optional, but helps to prevent a VM from automatically grabbing the YubiKey from another VM, or the host itself, when the YubiKey is inserted. The remaining lines enable direct passthrough of the YubiKey to the guest VM.*
5. Power on the VM and once it boots up check if you are getting the option to connect the YubiKey in dedicated mode. If the changes are done correctly, you should now see an option to connect the YubiKey in dedicated mode (non-shared).

Appendix F: Sample Token Selection Criteria Checklist Form

A sample token selection criteria checklist form is provided here for reference. It includes excerpts from Cisco's evaluation of YubiKey NEO and YubiKey 4 Series tokens. Note that this evaluation was conducted in early 2018 and the results are based on the features available at that time.

Self-Provisioning Related Requirements

Requirement	Finding
Support for PIV/Smartcard slots.	Supported
Ability to generate the private keys securely and directly on the hardware token.	Supported
Support for RSA 2048 or better.	Supported
Private keys generated on the hardware token should be NOT exportable.	Supported
Slot PIN policies should be NOT updatable once the key is generated.	Supported
Ability to attest and verify that the private key was generated on the hardware token.	Supported in YubiKey 4.3 and newer.
Ability to attest and verify that the PIN policies for the slot where the private key is generated.	Supported in YubiKey 4.3 and newer.
Ability to attest and verify the device SN, model/firmware version of the token on which the key/CSR is generated.	Supported
Ability to create a CSR using the private keys created in the hardware token.	Supported
Ability to import externally signed certificates to the slot that has the associated private key.	Supported
Support for multiple slots	Supported

Smartcard Logon Related Requirements

Requirement	Finding
Smartcard Logon to Remote Windows Servers using Remote Desktop Client	Works only in Windows. The Remote Desktop Client for Mac doesn't support Smartcard logon.
Smartcard pairing to local user account on Mac	Supported
Smartcard pairing to local user account on Windows	Supported

TLS Certificate Based Authentication Related Requirements

Requirement	Finding
Certificate Based Authentication to Websites on MacOS	Works in Safari, Chrome and Firefox with OpenSC installed. In Firefox OpenSC PKCS11 module should be loaded for this to work.
Certificate Based Authentication to Websites on Windows	Supported

SMIME Email Signing using the Certificate in YubiKey

Requirement	Finding
Email Signing on MacOS	Works in Microsoft Outlook with OpenSC installed.
Email Signing on Windows	Works in Microsoft Outlook