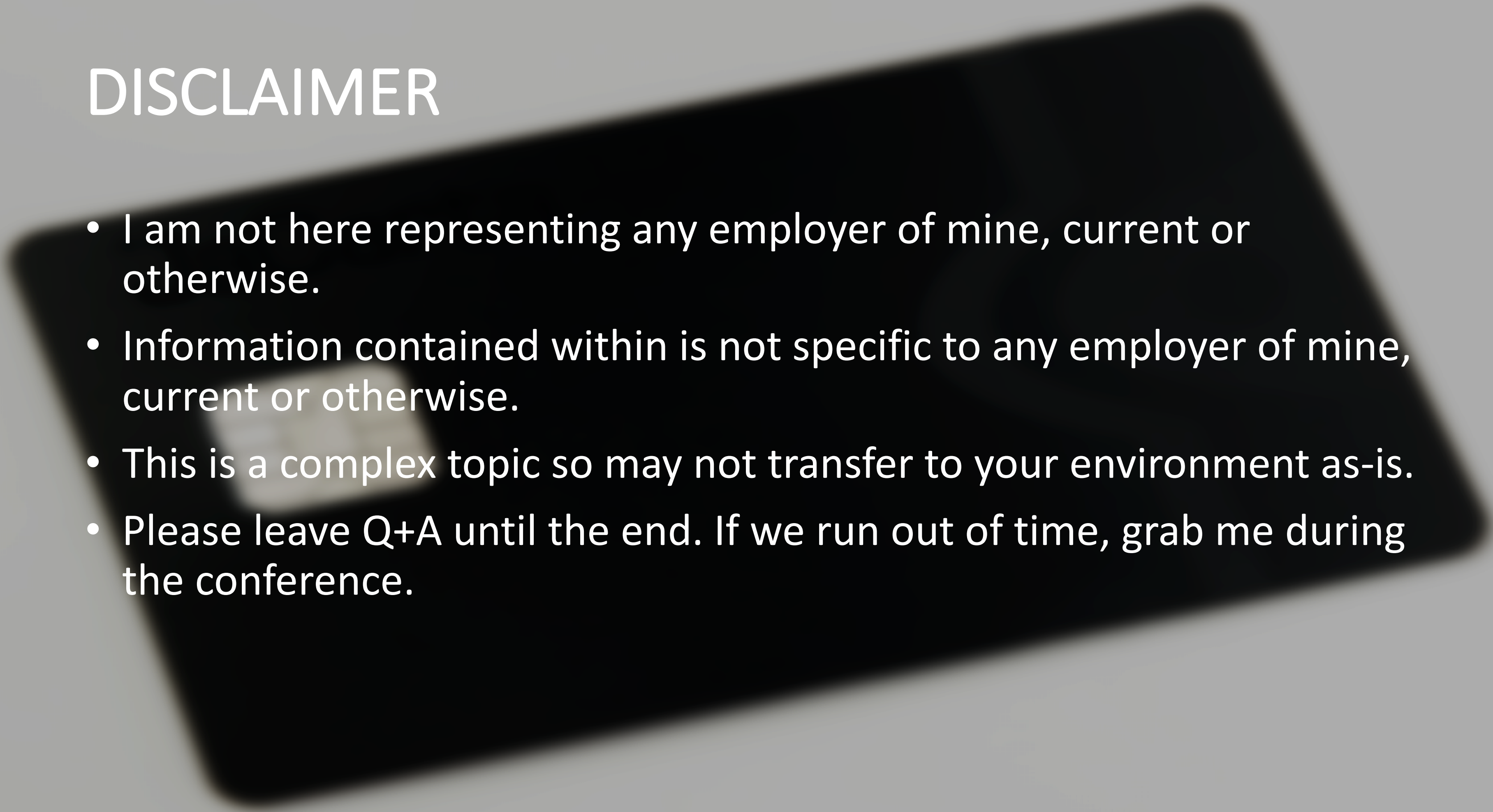
Working with Smart Cards: macOS and Security

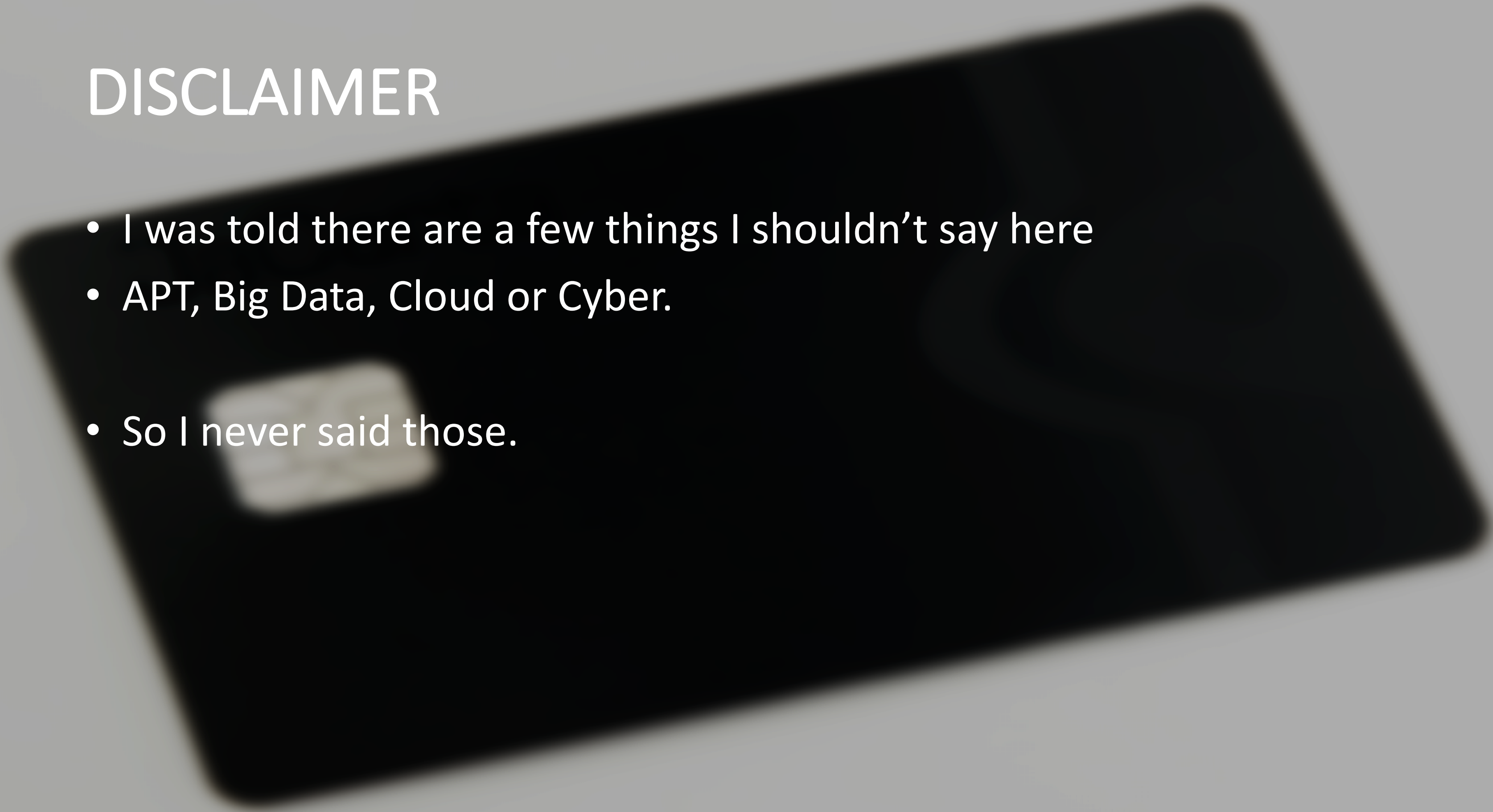
2017 MACADMINS CONFERENCE AT PENN STATE®

DISCLAIMER

DISCLAIMER



- I am not here representing any employer of mine, current or otherwise.
- Information contained within is not specific to any employer of mine, current or otherwise.
- This is a complex topic so may not transfer to your environment as-is.
- Please leave Q+A until the end. If we run out of time, grab me during the conference.



DISCLAIMER

- I was told there are a few things I shouldn’t say here
- APT, Big Data, Cloud or Cyber.

- So I never said those.

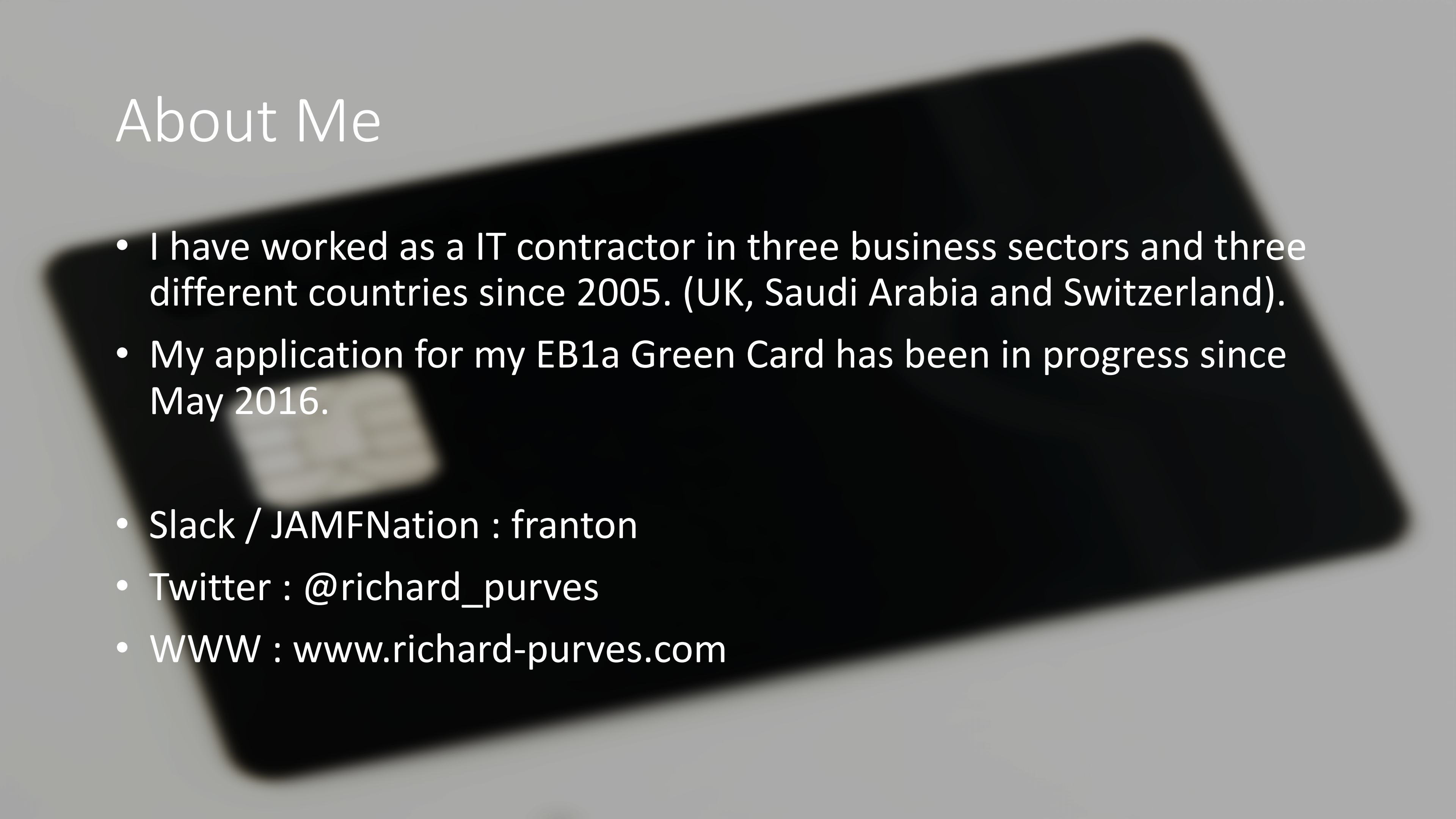
CHOCOLATE!

I’ve brought some Swiss chocolate with me.

Come down the front at any time and help yourself to one

Or two.

Or three.

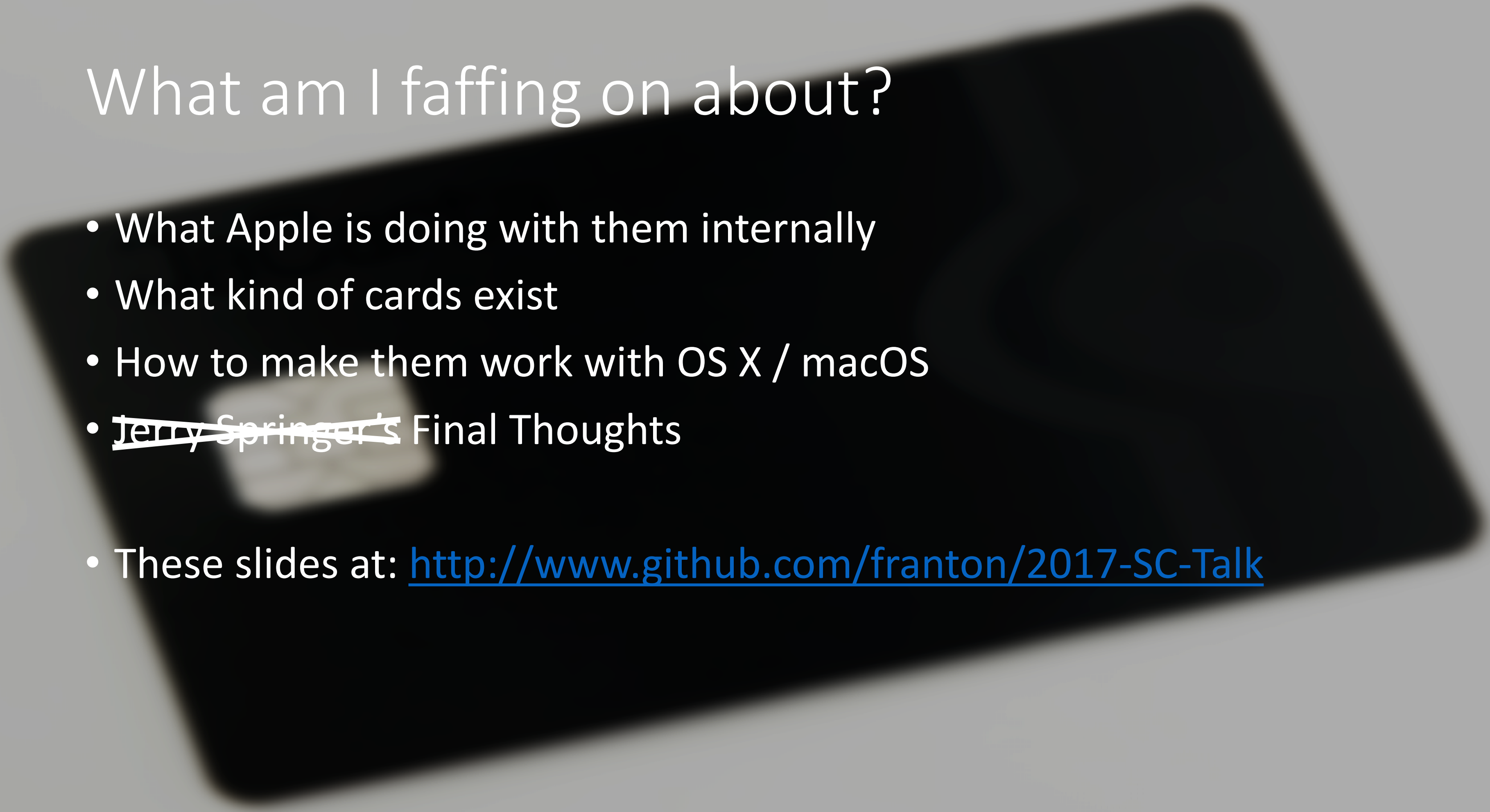


About Me

- I have worked as a IT contractor in three business sectors and three different countries since 2005. (UK, Saudi Arabia and Switzerland).
- My application for my EB1a Green Card has been in progress since May 2016.

- Slack / JAMFNation : franton
- Twitter : @richard_purves
- WWW : www.richard-purves.com

What am I faffing on about?



What am I faffing on about?

- What Apple is doing with them internally
- What kind of cards exist
- How to make them work with OS X / macOS
- ~~Jerry Springer's~~ Final Thoughts

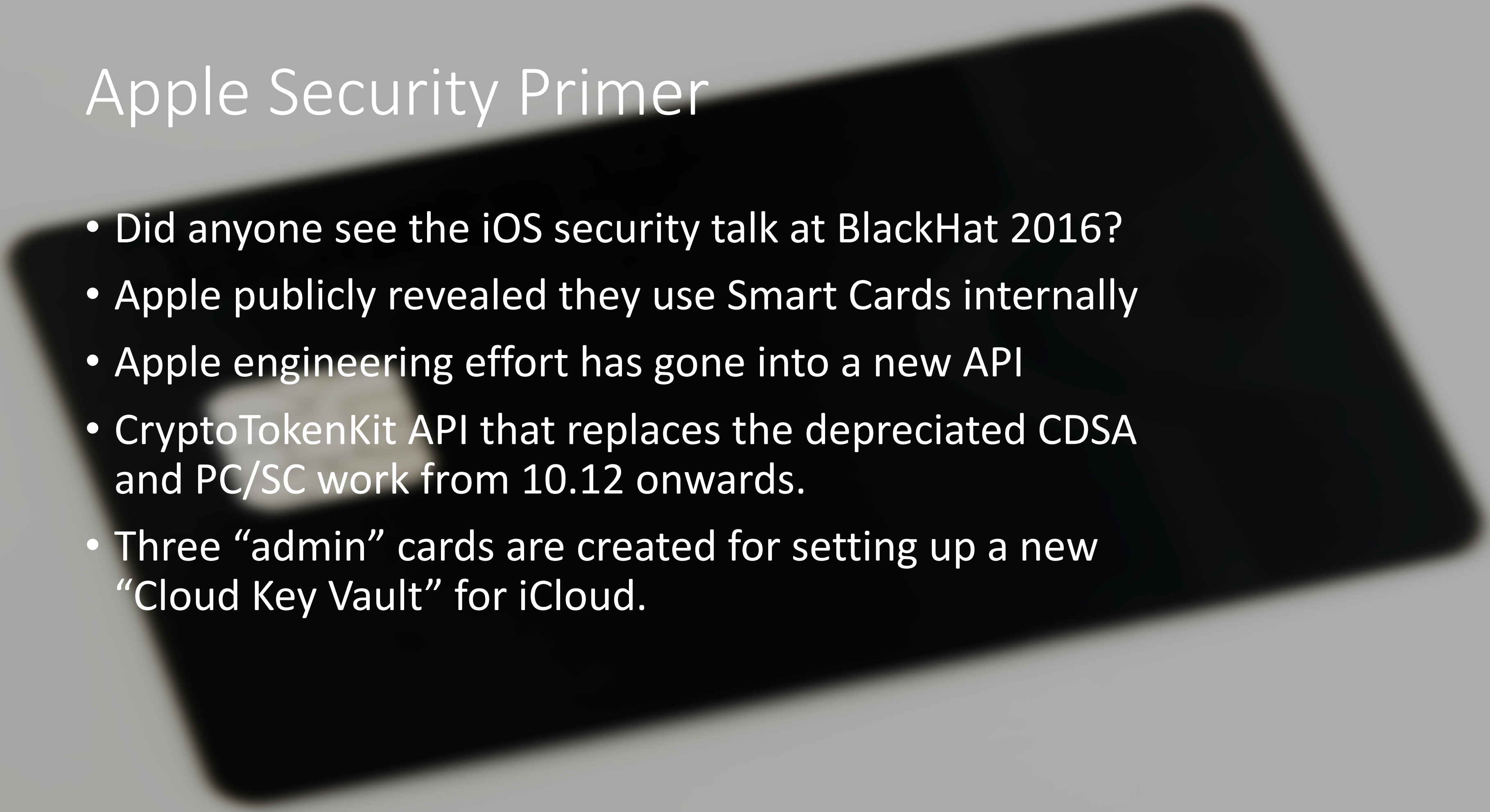
- These slides at: http://www.github.com/franton/2017-SC-Talk

Security Primer

Apple Security Primer

- Apple started back in 10.4 to use the following standards:
  - Common Data Security Architecture (CDSA)
  - Personal Computer / Smart Card (PC/SC)
- Apple marked the previous API’s and technologies in OS X as depreciated since 10.7
- Sat on macosxforge as open source until September 2016.
- Now on GitHub instead.

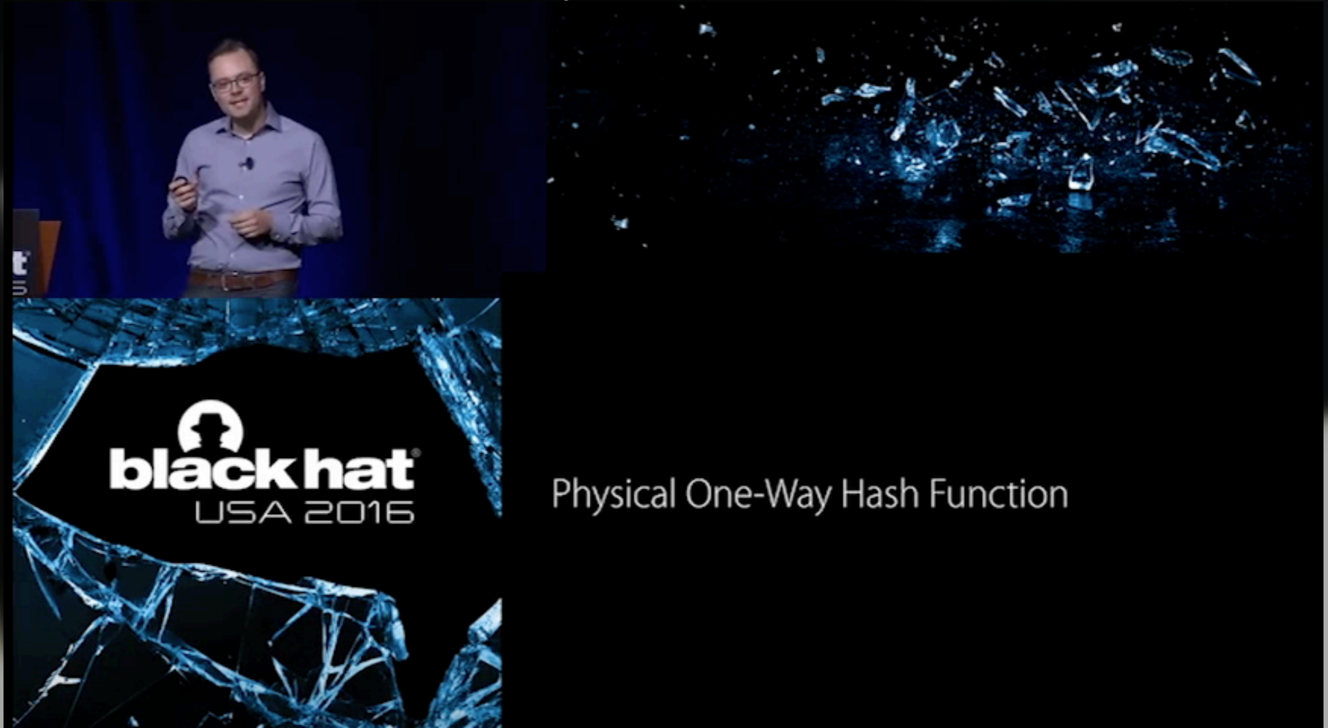
Apple Security Primer



- Did anyone see the iOS security talk at BlackHat 2016?
- Apple publicly revealed they use Smart Cards internally
- Apple engineering effort has gone into a new API
- CryptoTokenKit API that replaces the depreciated CDSA and PC/SC work from 10.12 onwards.
- Three “admin” cards are created for setting up a new “Cloud Key Vault” for iCloud.

Apple Security Primer

- Let’s hear from Ivan what they do next …



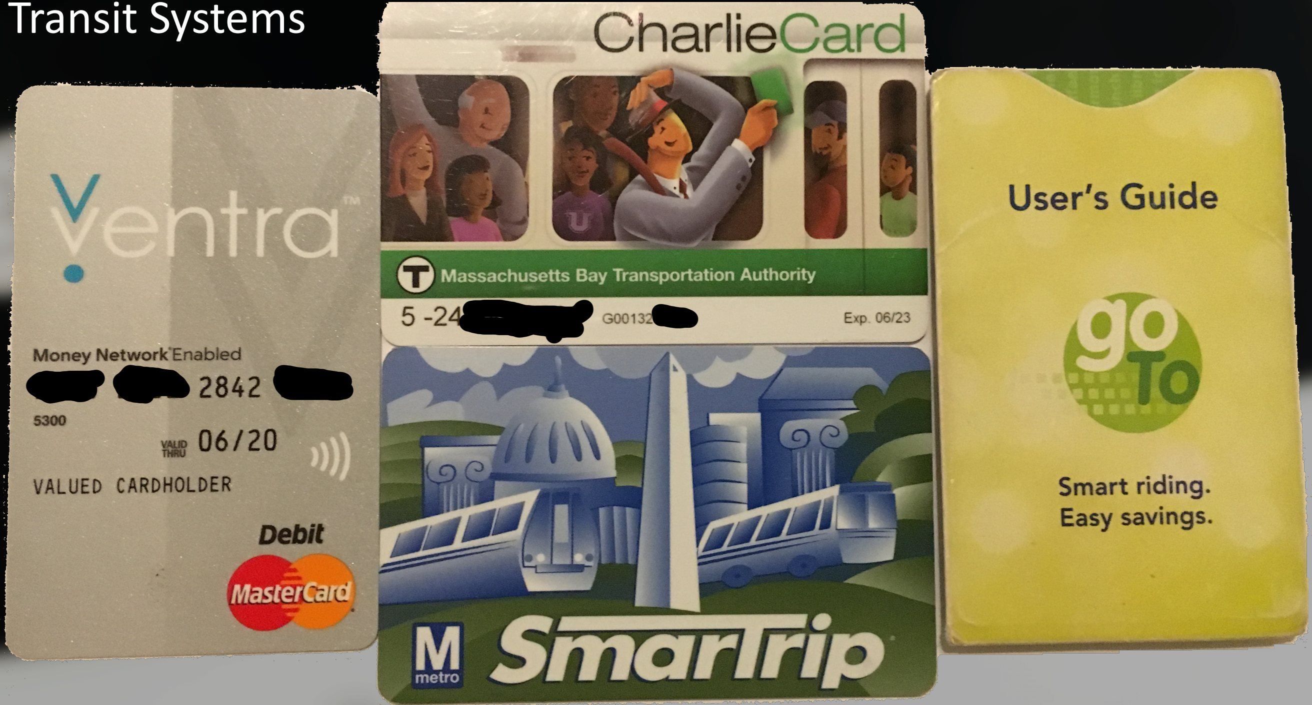
Enough! What are smart cards?

What are Smart Cards?

- Single Purpose and Multi-Purpose
  - What’s the difference?

What are Smart Cards?

- Single Purpose Cards
  - Public Transit Systems

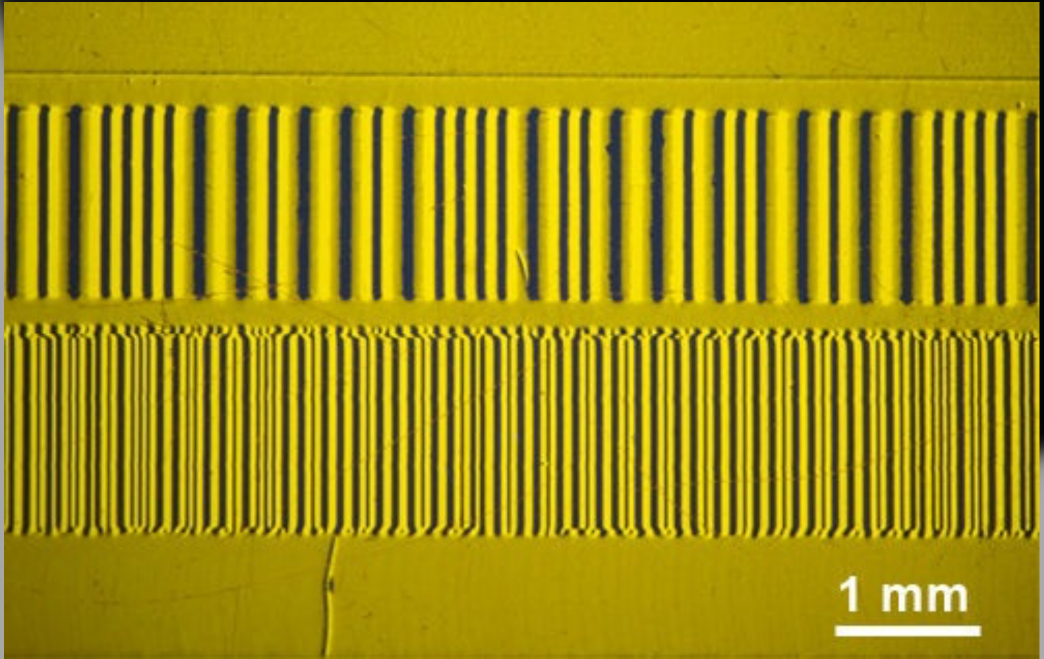


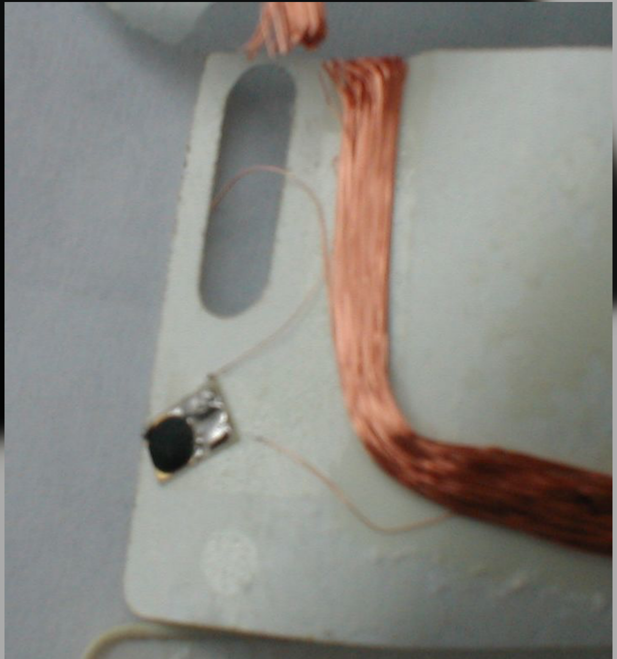
What are Smart Cards?

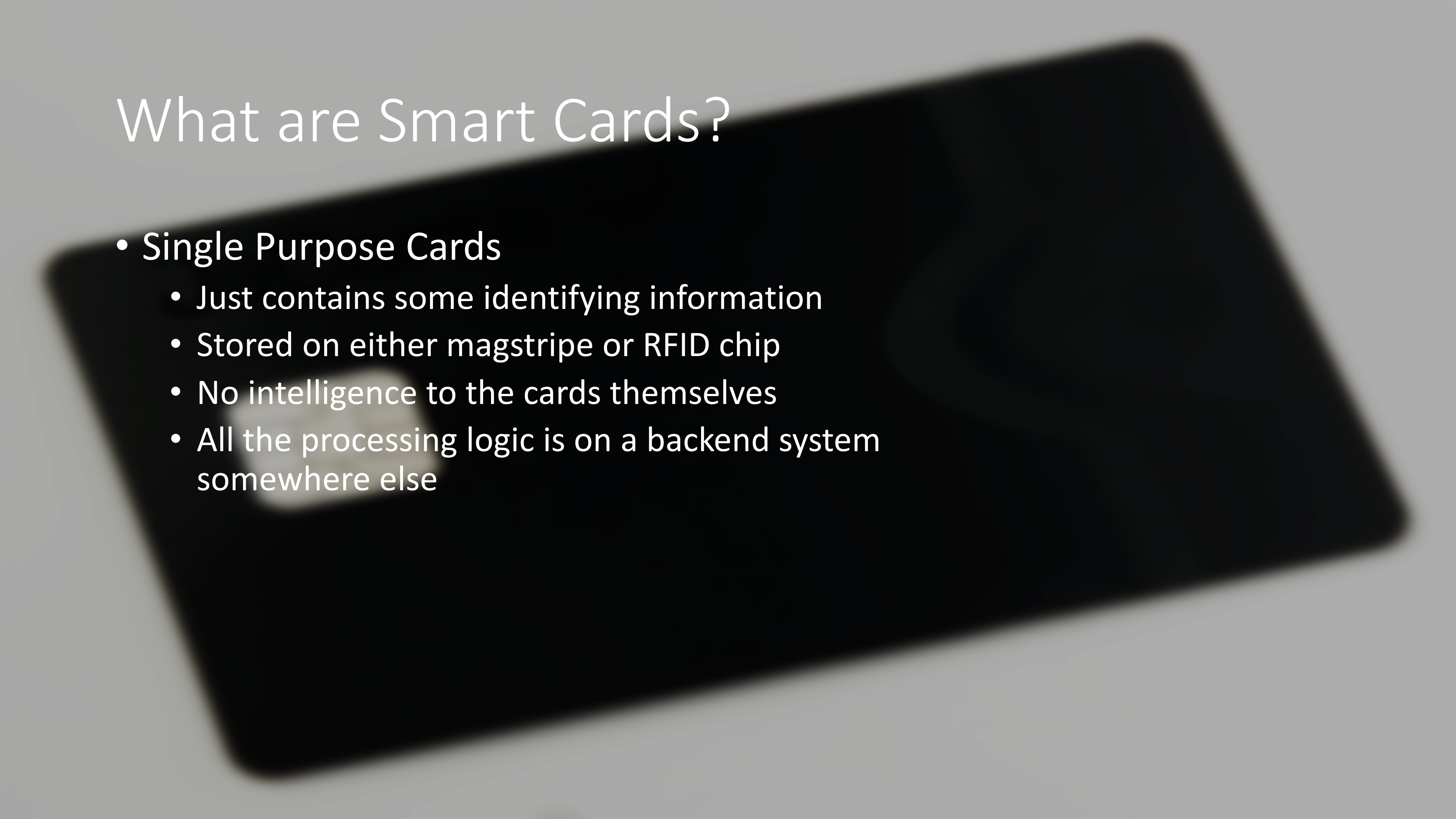
- Single Purpose Cards
  - Hotel Keycards
  - Loyalty Cards

What are Smart Cards?

- Single Purpose Cards
  - Magnetic Stripe & Contactless





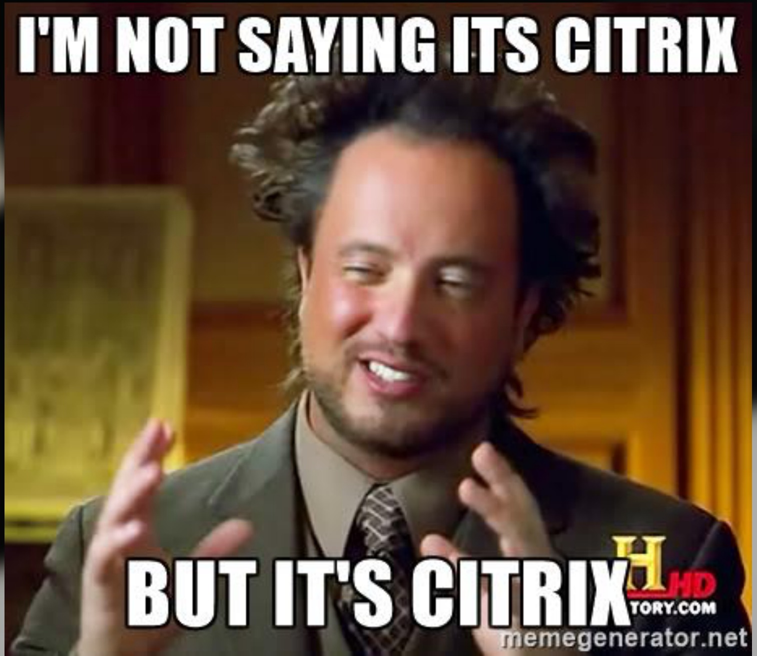


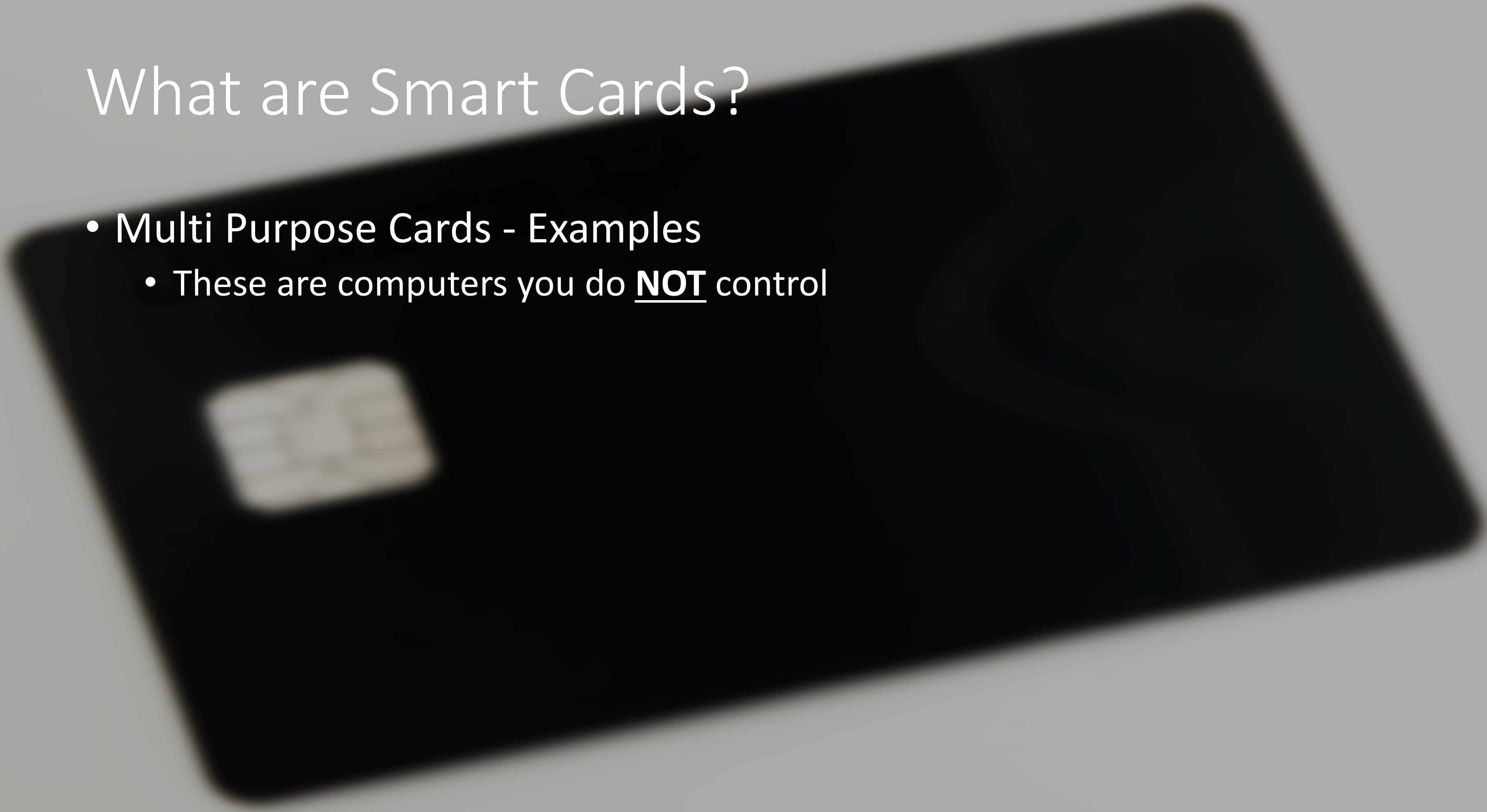
What are Smart Cards?

- Single Purpose Cards
  - Just contains some identifying information
  - Stored on either magstripe or RFID chip
  - No intelligence to the cards themselves
  - All the processing logic is on a backend system somewhere else

What are Smart Cards?

- Single Purpose Cards
  - Yep, they're basically Citrix.



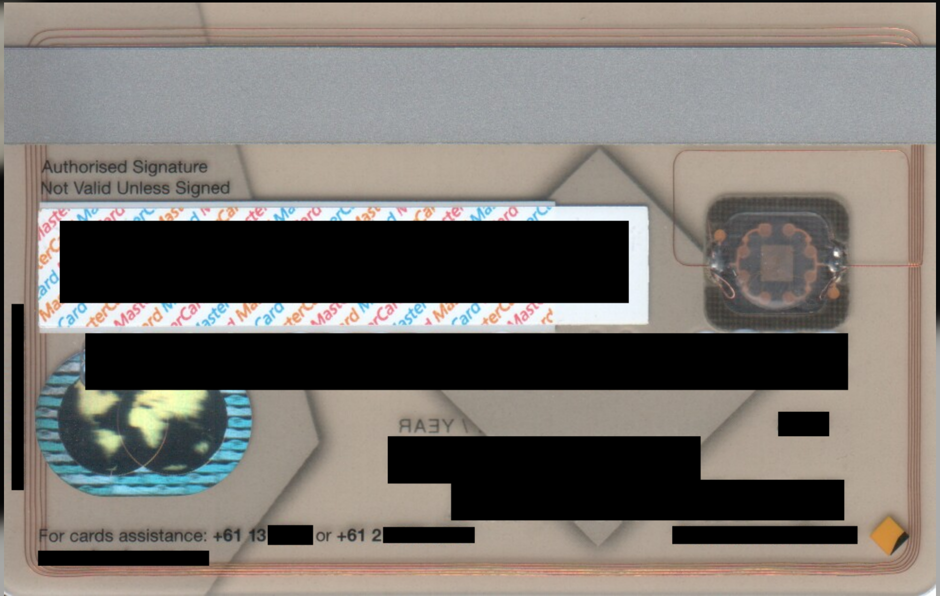


What are Smart Cards?

- Multi Purpose Cards - Examples
  - These are computers you do **NOT** control

What are Smart Cards?

- Multi Purpose Cards - Examples
  - These are computers you do **NOT** control
  - Debit / Credit Cards



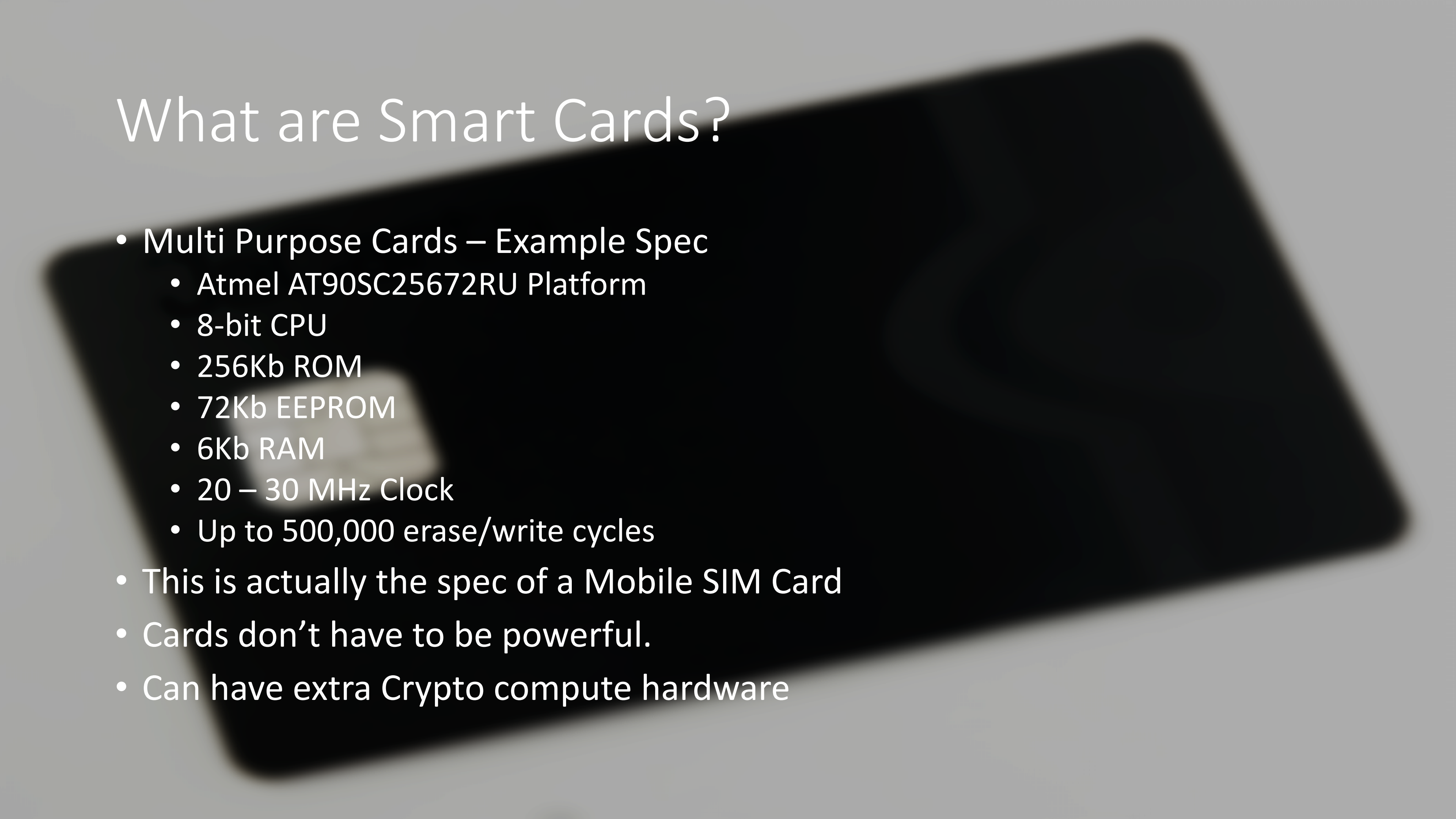
What are Smart Cards?

- Multi Purpose Cards - Examples
  - These are computers you do **NOT** control
  - Debit / Credit Cards
  - Subscriber Identity Module (SIM) Card

SIM Card
OPGELET!
Kijk voor u uw kaart losmaakt eerst na welk formaat bij uw toestel past.
ATTENTION!
Avant de détacher votre carte, vérifiez le format correspondant à votre appareil.
Mini
Micro
Nano
proximus

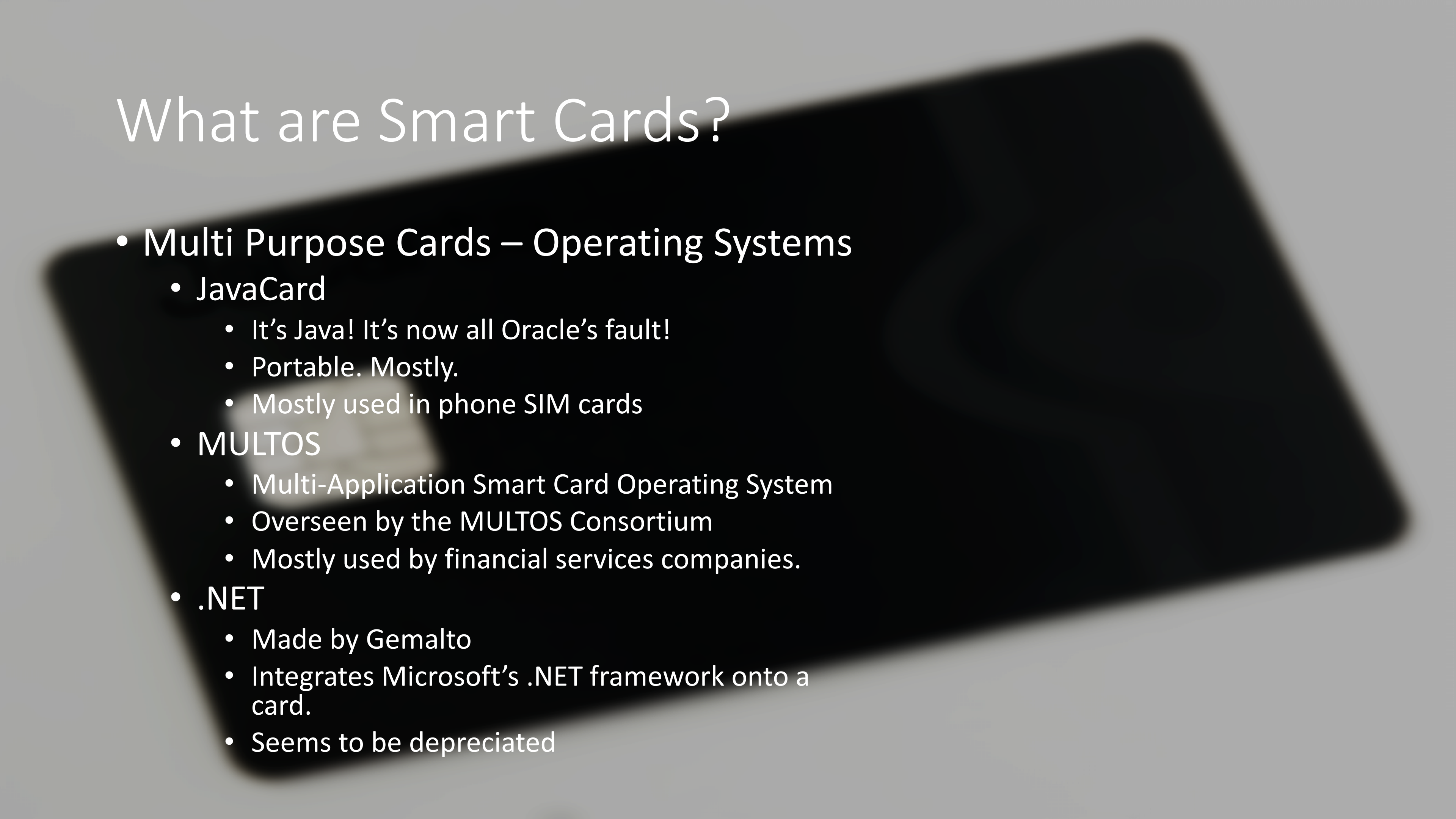
What are Smart Cards?

- Multi Purpose Cards - Examples
  - These are computers you do **NOT** control
  - Debit / Credit Cards
  - Subscriber Identity Module (SIM) Card
  - Computer Cards



What are Smart Cards?

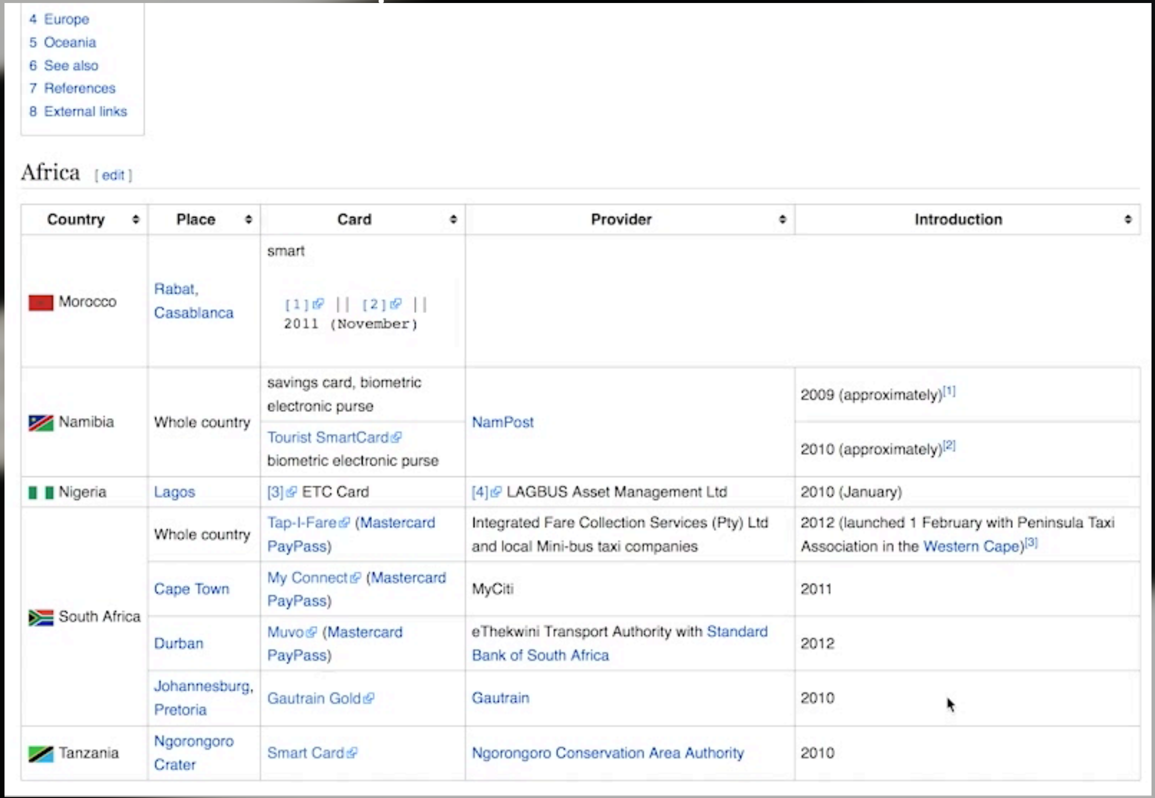
- Multi Purpose Cards – Example Spec
  - Atmel AT90SC25672RU Platform
  - 8-bit CPU
  - 256Kb ROM
  - 72Kb EEPROM
  - 6Kb RAM
  - 20 – 30 MHz Clock
  - Up to 500,000 erase/write cycles
- This is actually the spec of a Mobile SIM Card
- Cards don’t have to be powerful.
- Can have extra Crypto compute hardware



What are Smart Cards?

- Multi Purpose Cards – Operating Systems
  - JavaCard
    - It’s Java! It’s now all Oracle’s fault!
    - Portable. Mostly.
    - Mostly used in phone SIM cards
  - MULTOS
    - Multi-Application Smart Card Operating System
    - Overseen by the MULTOS Consortium
    - Mostly used by financial services companies.
  - .NET
    - Made by Gemalto
    - Integrates Microsoft’s .NET framework onto a card.
    - Seems to be depreciated

Smart Card Examples



Africa [edit]

Country	Place	Card	Provider	Introduction
Morocco	Rabat, Casablanca	smart [1] || [2] || 2011 (November)		
Namibia	Whole country	savings card, biometric electronic purse	NamPost	2009 (approximately)[1]
		Tourist SmartCard biometric electronic purse		2010 (approximately)[2]
Nigeria	Lagos	[3] ETC Card	[4] LAGBUS Asset Management Ltd	2010 (January)
South Africa	Whole country	Tap-I-Fare (Mastercard PayPass)	Integrated Fare Collection Services (Pty) Ltd and local Mini-bus taxi companies	2012 (launched 1 February with Peninsula Taxi Association in the Western Cape)[3]
	Cape Town	My Connect (Mastercard PayPass)	MyCiti	2011
	Durban	Muvo (Mastercard PayPass)	eThekwini Transport Authority with Standard Bank of South Africa	2012
	Johannesburg, Pretoria	Gautrain Gold	Gautrain	2010
Tanzania	Ngorongoro Crater	Smart Card	Ngorongoro Conservation Area Authority	2010

(with thanks to Mr. Regular of "Regular Car Reviews")

Making them work for you

How to make them work

- USB Card Reader - Omnikey 3121

How to make them work

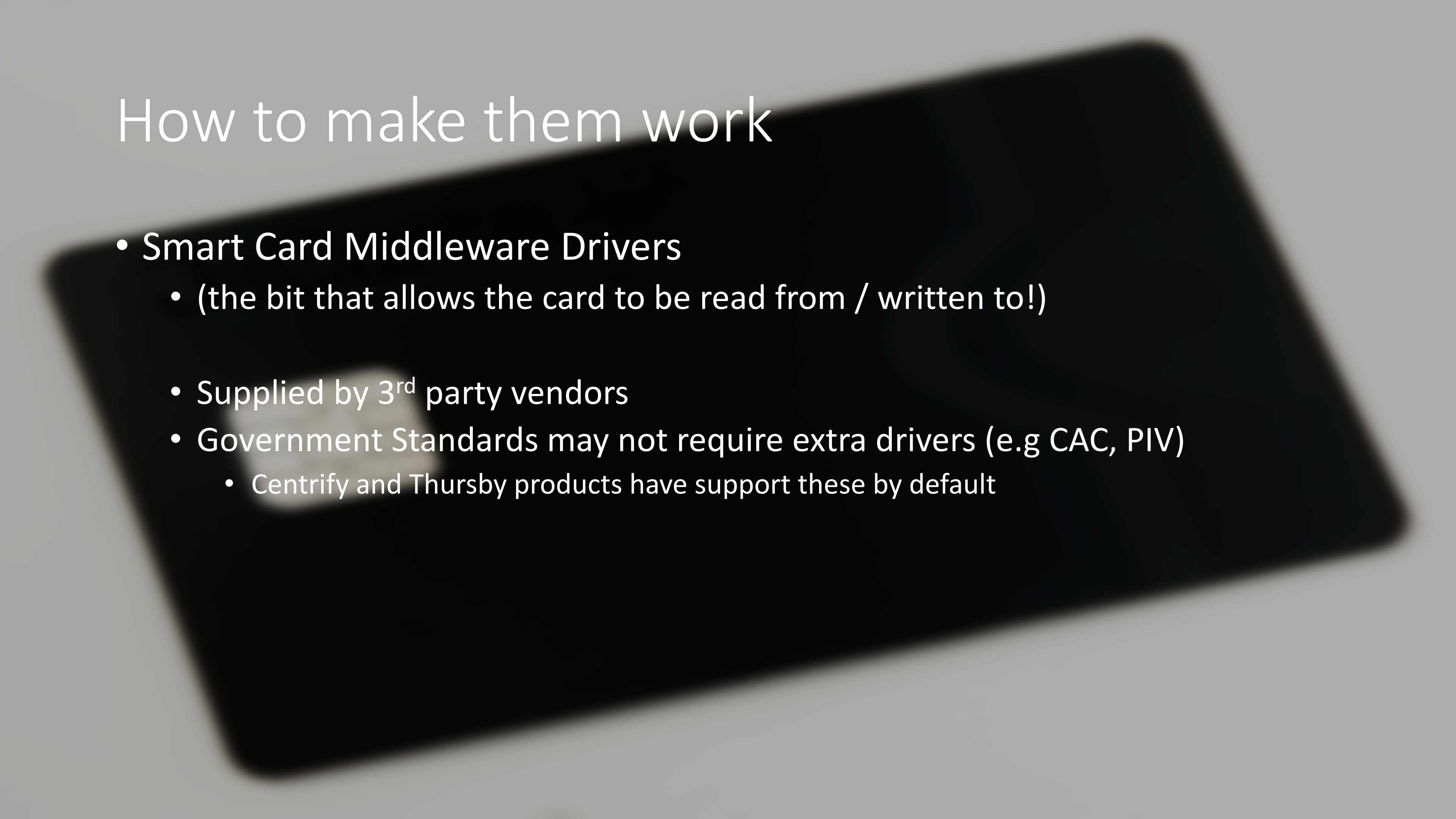
- USB Card Reader – Identive SCR3500

How to make them work

- USB Card Reader – Identive SCR3500A

How to make them work

- USB Card Reader – Identive Cloud 2900R



How to make them work

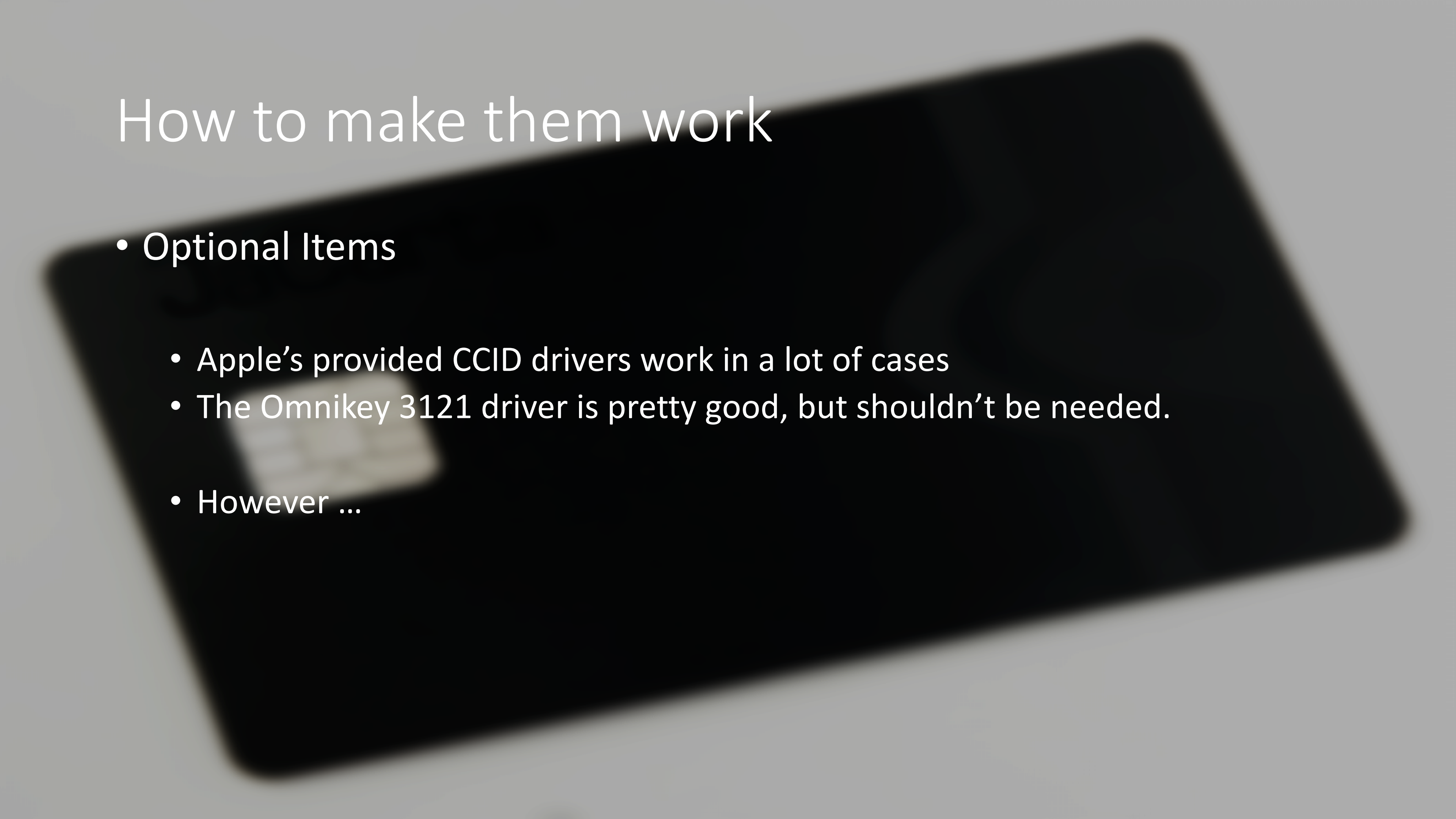
- Smart Card Middleware Drivers
    - (the bit that allows the card to be read from / written to!)

    - Supplied by 3rd party vendors
    - Government Standards may not require extra drivers (e.g CAC, PIV)
        - Centrify and Thursby products have support these by default

How to make them work

- Loginwindow Configuration
  - Make sure the loginwindow is set to display as “List of users”.
  - If set to username/password, none of this works

• PayloadType:	com.apple.loginwindow
• Key:	SHOWFULLNAME
• Type:	integer
• Setting:	0



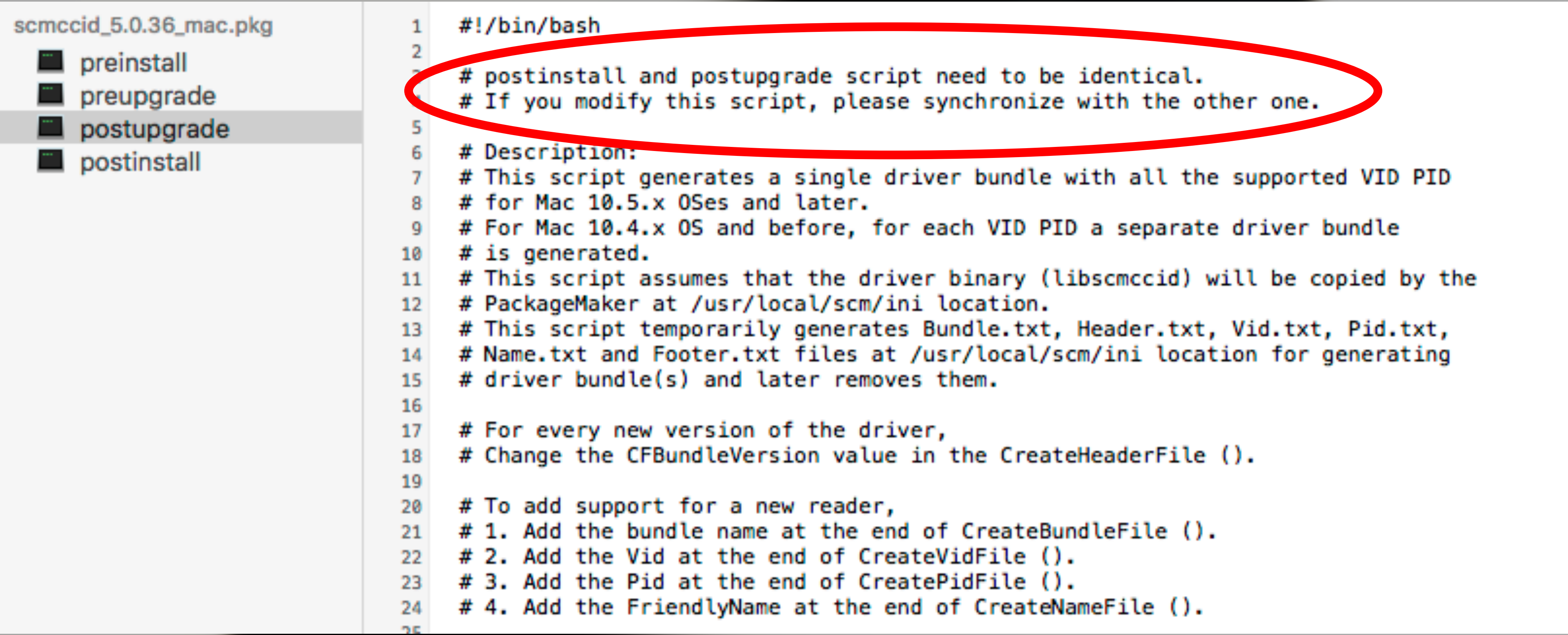
How to make them work

- Optional Items
  - Apple’s provided CCID drivers work in a lot of cases
  - The Omnikey 3121 driver is pretty good, but shouldn’t be needed.

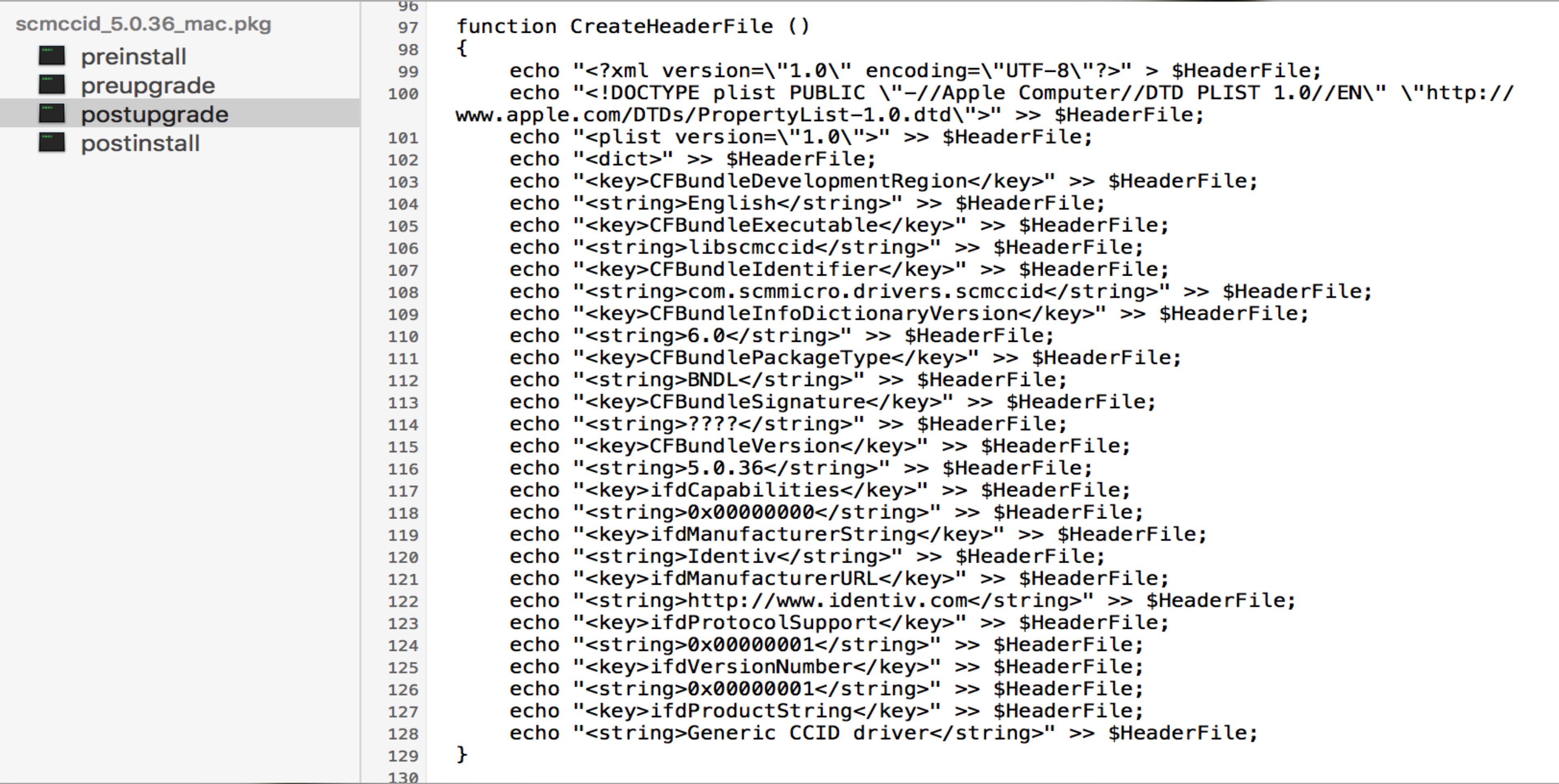
  - However …

How to make them work

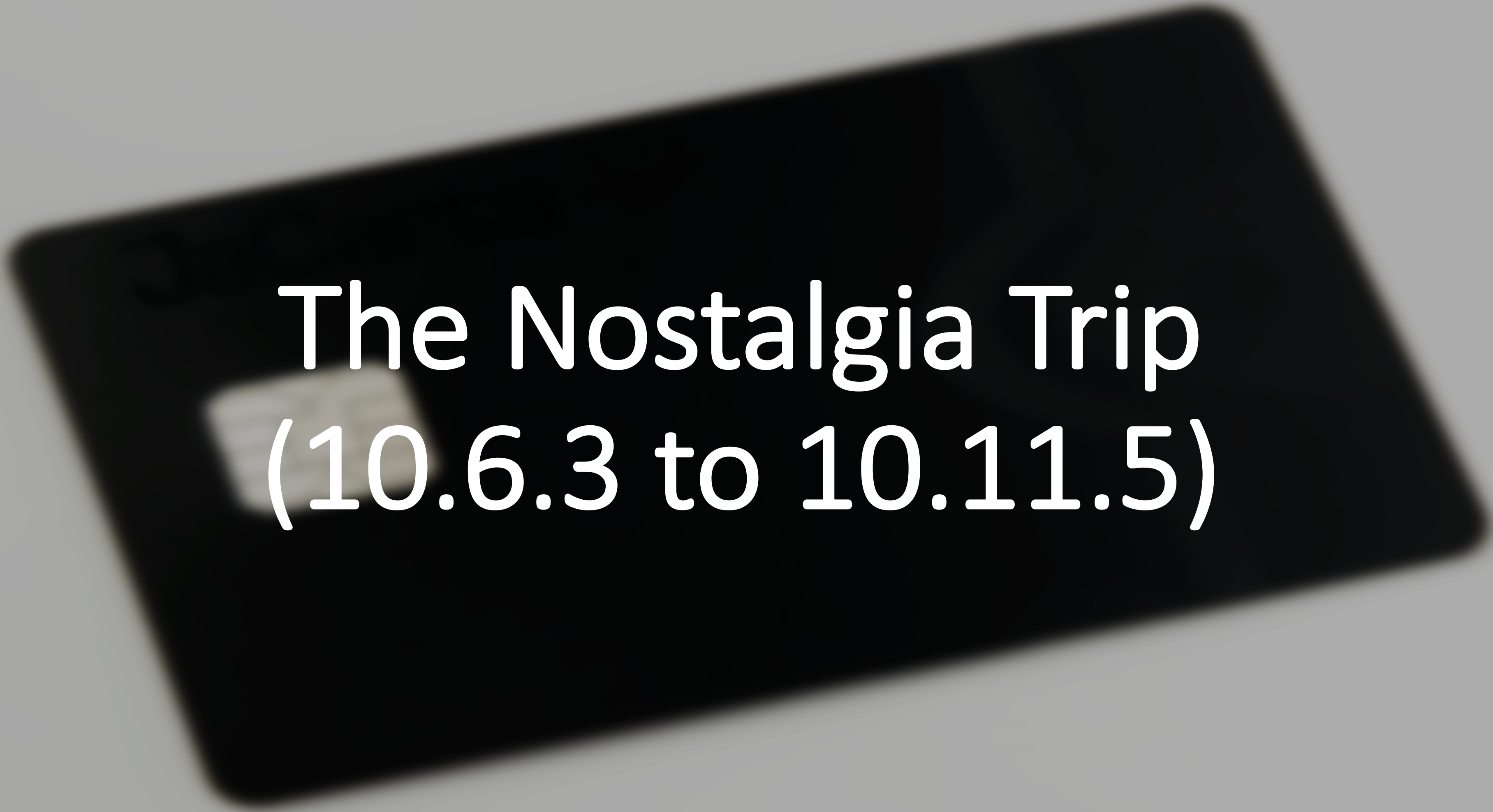
- (optional) Smart Card USB Driver
- Identiv SCR3500A / Cloud 2900R
  - One driver fits all
  - Contender for “Most inept macOS package ever made”



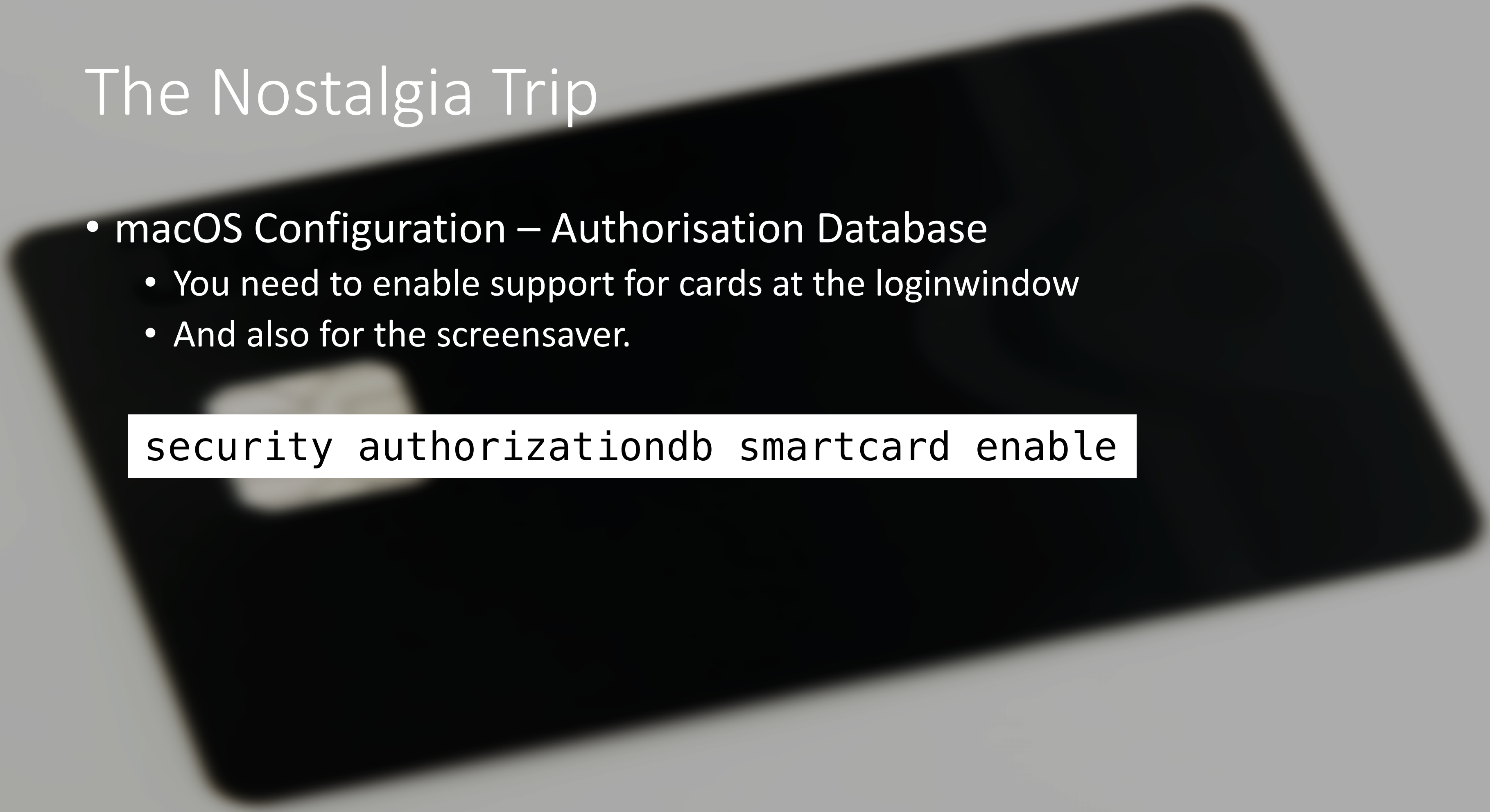
```
#!/bin/bash

# postinstall and postupgrade script need to be identical.
# If you modify this script, please synchronize with the other one.

# Description:
# This script generates a single driver bundle with all the supported VID PID
# for Mac 10.5.x OSes and later.
# For Mac 10.4.x OS and before, for each VID PID a separate driver bundle
# is generated.
# This script assumes that the driver binary (libscmccid) will be copied by the
# PackageMaker at /usr/local/scm/ini location.
# This script temporarily generates Bundle.txt, Header.txt, Vid.txt, Pid.txt,
# Name.txt and Footer.txt files at /usr/local/scm/ini location for generating
# driver bundle(s) and later removes them.

# For every new version of the driver,
# Change the CFBundleVersion value in the CreateHeaderFile ().

# To add support for a new reader,
# 1. Add the bundle name at the end of CreateBundleFile ().
# 2. Add the Vid at the end of CreateVidFile ().
# 3. Add the Pid at the end of CreatePidFile ().
# 4. Add the FriendlyName at the end of CreateNameFile ().
```



```
function CreateHeaderFile ()
{
    echo "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" > $HeaderFile;
    echo "<!DOCTYPE plist PUBLIC \"-//Apple Computer//DTD PLIST 1.0//EN\" \"http://www.apple.com/DTDs/PropertyList-1.0.dtd\">" >> $HeaderFile;
    echo "<plist version=\"1.0\">" >> $HeaderFile;
    echo "<dict>" >> $HeaderFile;
    echo "<key>CFBundleDevelopmentRegion</key>" >> $HeaderFile;
    echo "<string>English</string>" >> $HeaderFile;
    echo "<key>CFBundleExecutable</key>" >> $HeaderFile;
    echo "<string>libscmccid</string>" >> $HeaderFile;
    echo "<key>CFBundleIdentifier</key>" >> $HeaderFile;
    echo "<string>com.scmmicro.drivers.scmccid</string>" >> $HeaderFile;
    echo "<key>CFBundleInfoDictionaryVersion</key>" >> $HeaderFile;
    echo "<string>6.0</string>" >> $HeaderFile;
    echo "<key>CFBundlePackageType</key>" >> $HeaderFile;
    echo "<string>BNDL</string>" >> $HeaderFile;
    echo "<key>CFBundleSignature</key>" >> $HeaderFile;
    echo "<string>????</string>" >> $HeaderFile;
    echo "<key>CFBundleVersion</key>" >> $HeaderFile;
    echo "<string>5.0.36</string>" >> $HeaderFile;
    echo "<key>ifdCapabilities</key>" >> $HeaderFile;
    echo "<string>0x00000000</string>" >> $HeaderFile;
    echo "<key>ifdManufacturerString</key>" >> $HeaderFile;
    echo "<string>Identiv</string>" >> $HeaderFile;
    echo "<key>ifdManufacturerURL</key>" >> $HeaderFile;
    echo "<string>http://www.identiv.com</string>" >> $HeaderFile;
    echo "<key>ifdProtocolSupport</key>" >> $HeaderFile;
    echo "<string>0x00000001</string>" >> $HeaderFile;
    echo "<key>ifdVersionNumber</key>" >> $HeaderFile;
    echo "<string>0x00000001</string>" >> $HeaderFile;
    echo "<key>ifdProductString</key>" >> $HeaderFile;
    echo "<string>Generic CCID driver</string>" >> $HeaderFile;
}
```



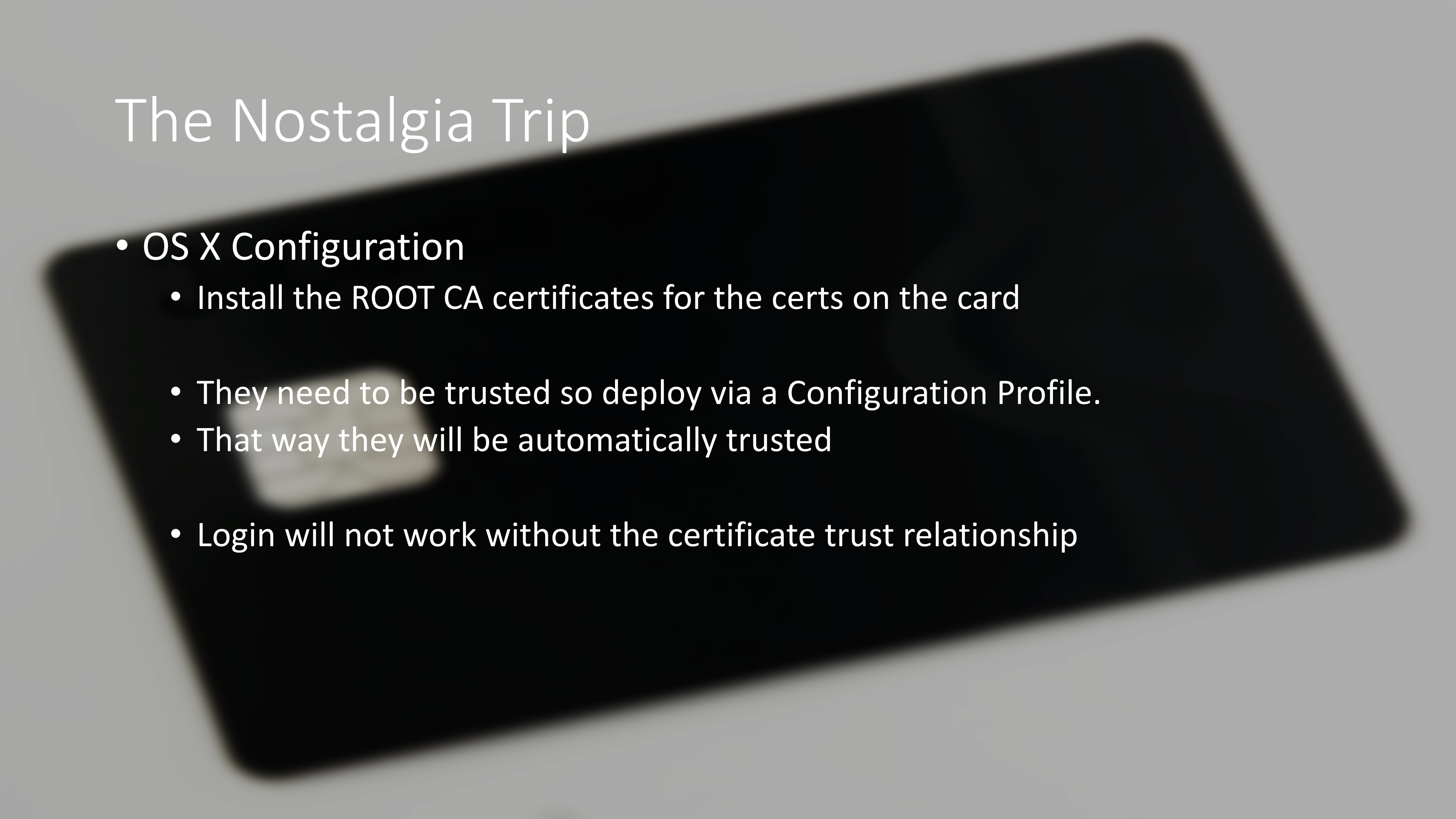
The Nostalgia Trip (10.6.3 to 10.11.5)



The Nostalgia Trip

- macOS Configuration – Authorisation Database
  - You need to enable support for cards at the loginwindow
  - And also for the screensaver.

```
security authorizationdb smartcard enable
```

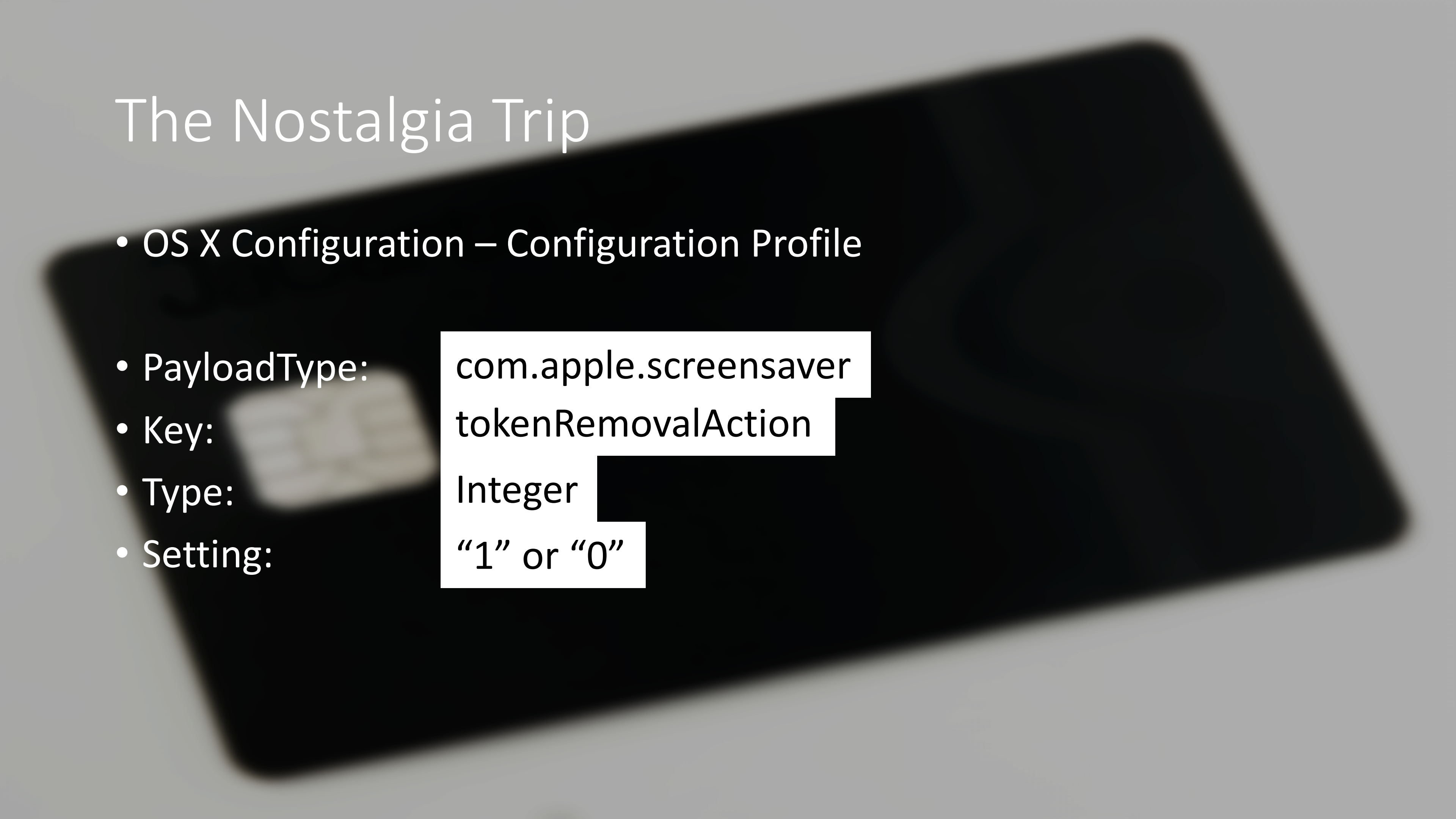


The Nostalgia Trip

- OS X Configuration
  - Install the ROOT CA certificates for the certs on the card

  - They need to be trusted so deploy via a Configuration Profile.
  - That way they will be automatically trusted

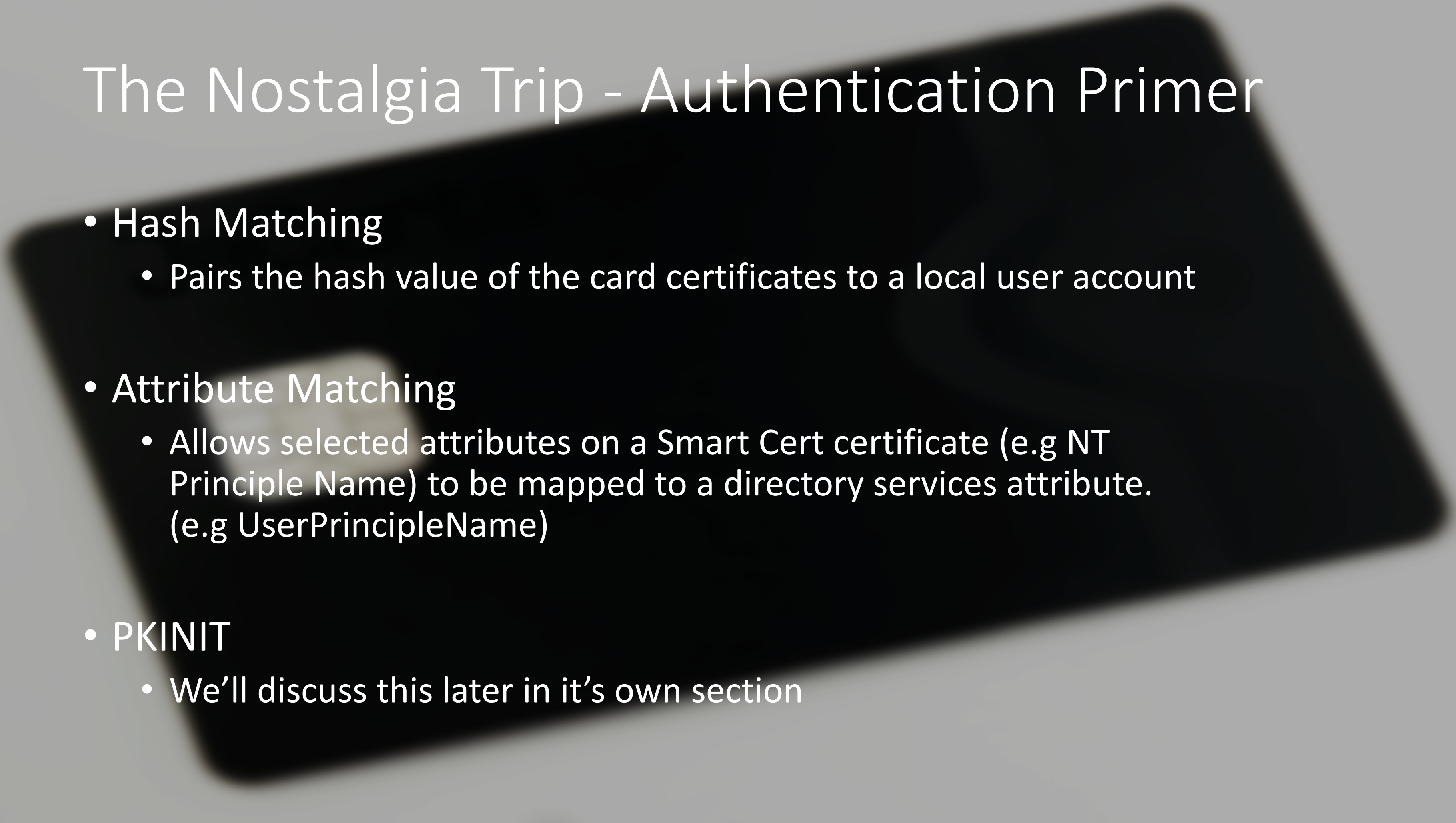
  - Login will not work without the certificate trust relationship



The Nostalgia Trip

- OS X Configuration – Configuration Profile

• PayloadType:	com.apple.screensaver
• Key:	tokenRemovalAction
• Type:	Integer
• Setting:	“1” or “0”



The Nostalgia Trip - Authentication Primer

- Hash Matching
  - Pairs the hash value of the card certificates to a local user account
- Attribute Matching
  - Allows selected attributes on a Smart Cert certificate (e.g NT Principle Name) to be mapped to a directory services attribute. (e.g UserPrincipleName)
- PKINIT
  - We'll discuss this later in it's own section

The Nostalgia Trip - Authentication Methods

- Hash Matching – sc_auth command (all OS versions)
  - (run as root!)

```
sc_auth hash
```

  - Lists the hashes of all certificates visible to the OS

```
sc_auth accept –u *username* –h *hash*
```

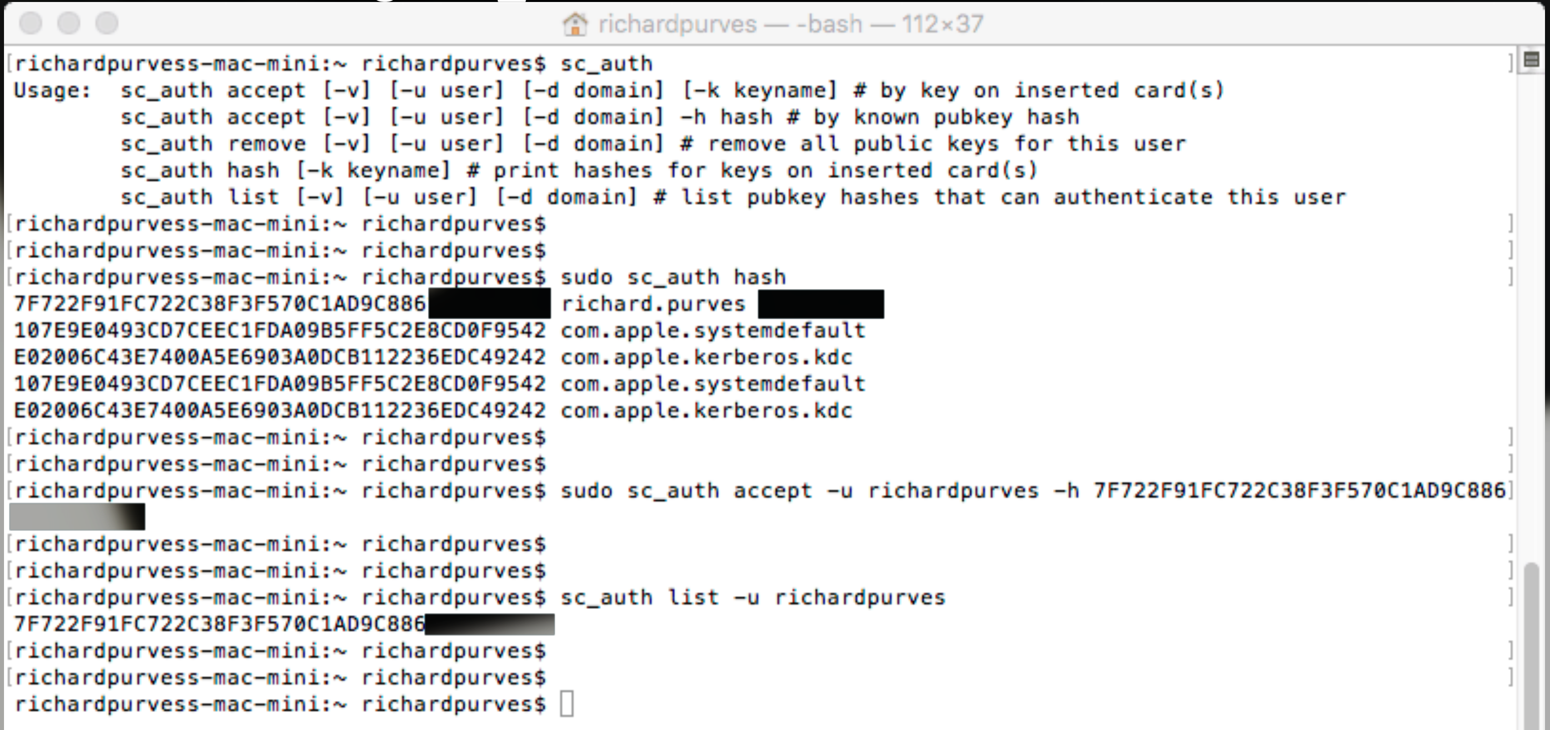
  - Adds the hash of the cert to the local user account

```
sc_auth list –u *username*
```

  - Shows any certificate hashes for a given username

The Nostalgia Trip - Authentication Methods

- Hash Matching – sc_auth command

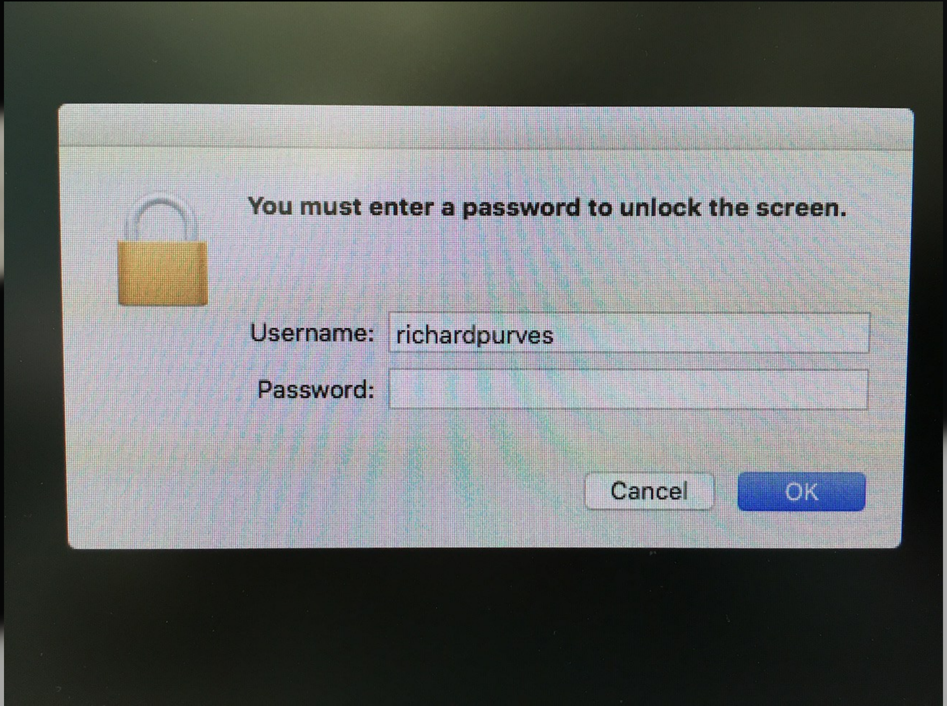


```
richardpurvess-mac-mini:~ richardpurves$ sc_auth
Usage:  sc_auth accept [-v] [-u user] [-d domain] [-k keyname] # by key on inserted card(s)
        sc_auth accept [-v] [-u user] [-d domain] -h hash # by known pubkey hash
        sc_auth remove [-v] [-u user] [-d domain] # remove all public keys for this user
        sc_auth hash [-k keyname] # print hashes for keys on inserted card(s)
        sc_auth list [-v] [-u user] [-d domain] # list pubkey hashes that can authenticate this user
richardpurvess-mac-mini:~ richardpurves$
richardpurvess-mac-mini:~ richardpurves$
richardpurvess-mac-mini:~ richardpurves$ sudo sc_auth hash
7F722F91FC722C38F3F570C1AD9C886[redacted] richard.purves [redacted]
107E9E0493CD7CEEC1FDA09B5FF5C2E8CD0F9542 com.apple.systemdefault
E02006C43E7400A5E6903A0DCB112236EDC49242 com.apple.kerberos.kdc
107E9E0493CD7CEEC1FDA09B5FF5C2E8CD0F9542 com.apple.systemdefault
E02006C43E7400A5E6903A0DCB112236EDC49242 com.apple.kerberos.kdc
richardpurvess-mac-mini:~ richardpurves$
richardpurvess-mac-mini:~ richardpurves$
richardpurvess-mac-mini:~ richardpurves$ sudo sc_auth accept -u richardpurves -h 7F722F91FC722C38F3F570C1AD9C886
[redacted]
richardpurvess-mac-mini:~ richardpurves$
richardpurvess-mac-mini:~ richardpurves$
richardpurvess-mac-mini:~ richardpurves$ sc_auth list -u richardpurves
7F722F91FC722C38F3F570C1AD9C886[redacted]
richardpurvess-mac-mini:~ richardpurves$
richardpurvess-mac-mini:~ richardpurves$
richardpurvess-mac-mini:~ richardpurves$
```

We’ll cover Attribute Matching in the PKINIT section

The Nostalgia Trip - Authentication Methods

- Local Account Login



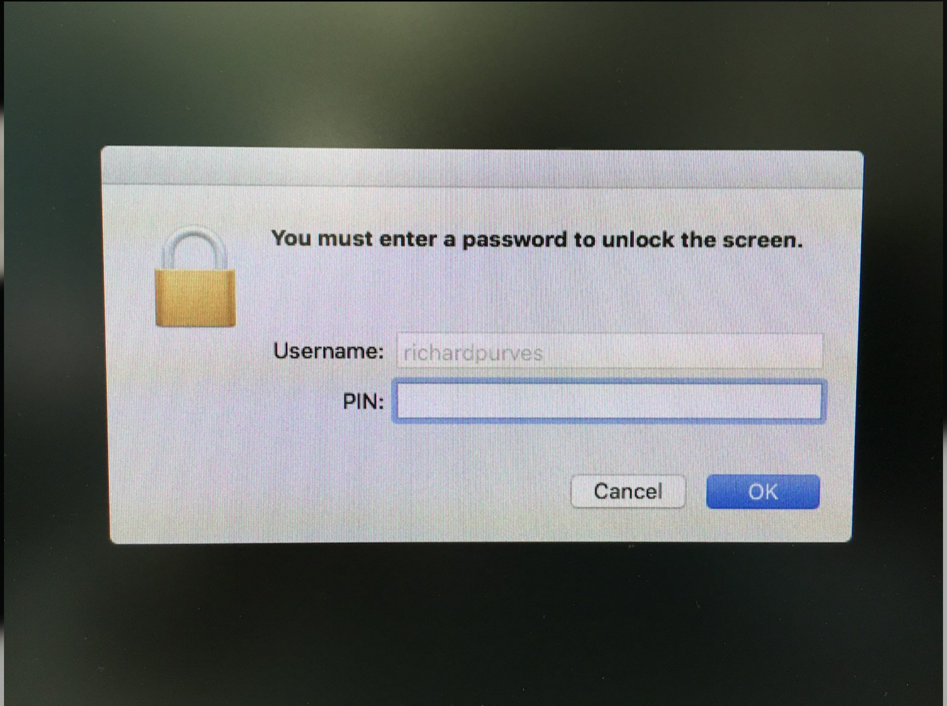
You must enter a password to unlock the screen.

Username: richardpurves

Password:

The Nostalgia Trip - Authentication Methods

- Local Account w/ Smartcard Login

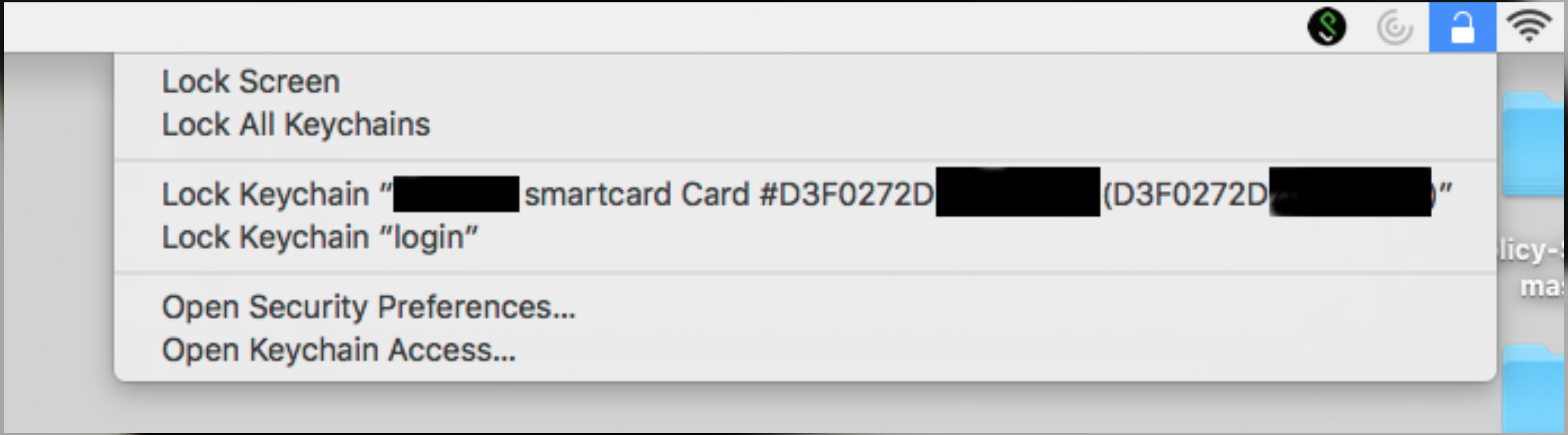

You must enter a password to unlock the screen.

Username: richardpurves

PIN:

The Nostalgia Trip

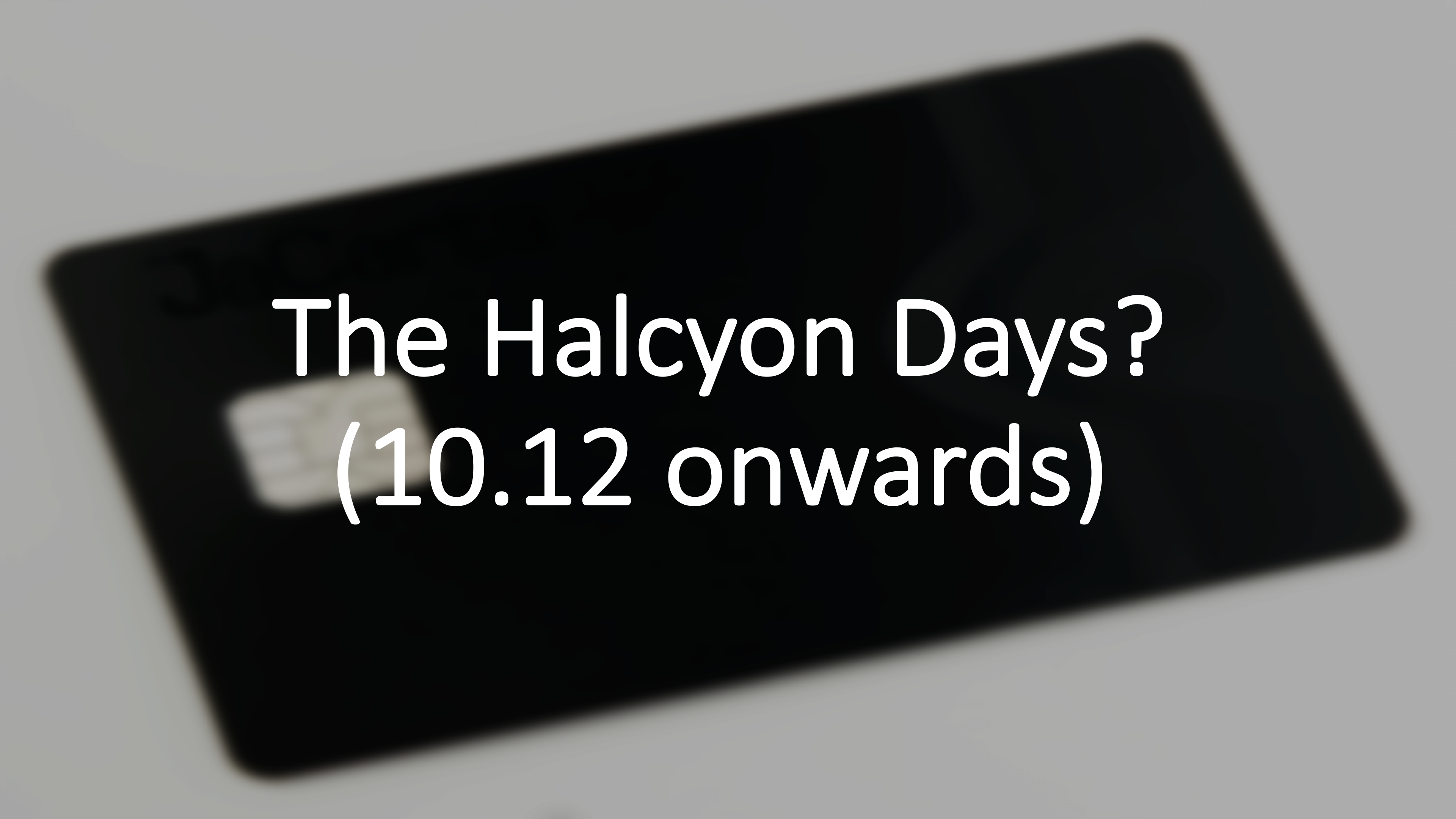
- And as if by magic …
  - Your smart card should appear as a secondary keychain on the system



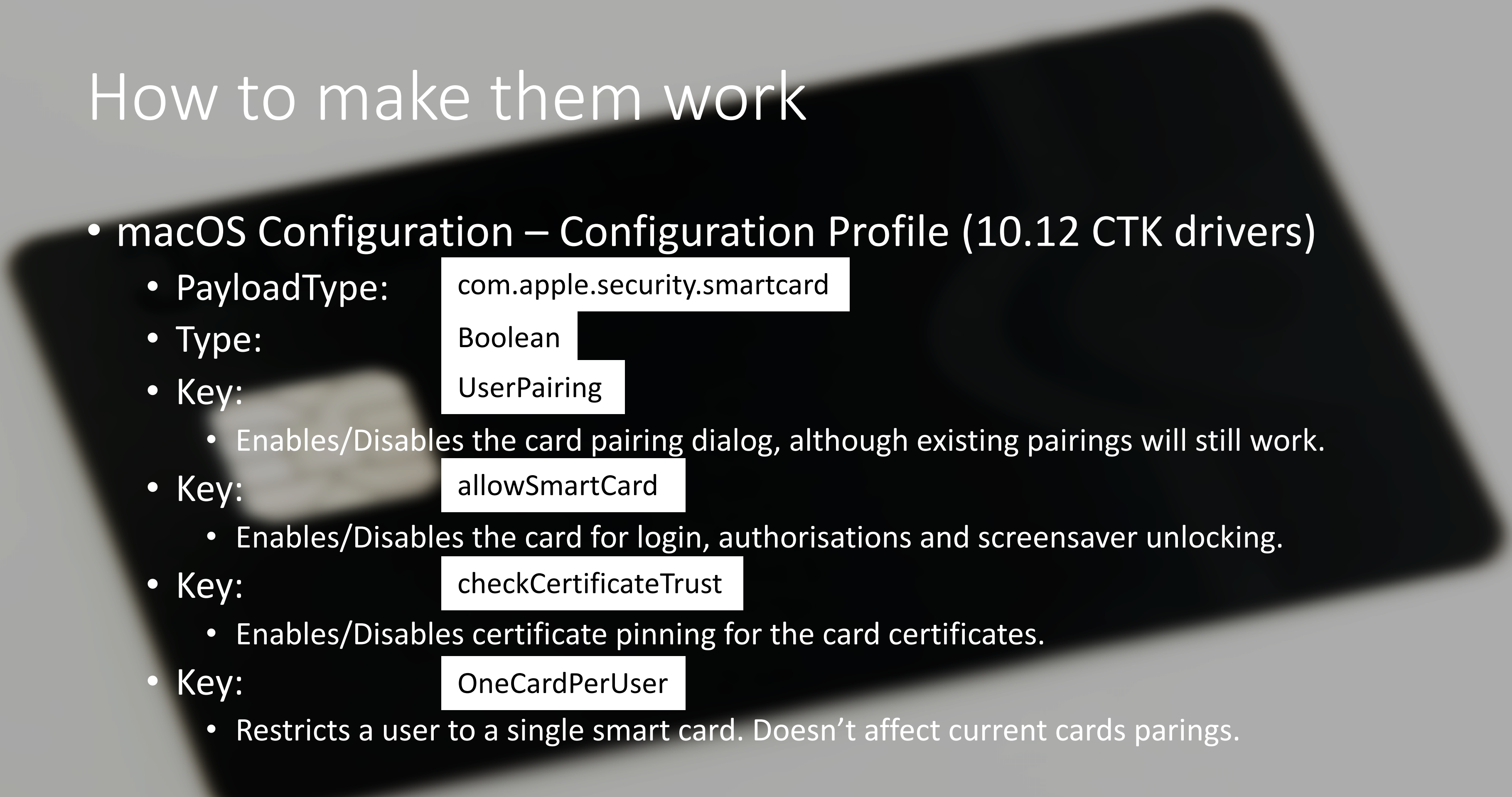
Lock Screen
Lock All Keychains

Lock Keychain "[redacted] smartcard Card #D3F0272D[redacted] (D3F0272D[redacted])"
Lock Keychain "login"

Open Security Preferences...
Open Keychain Access...

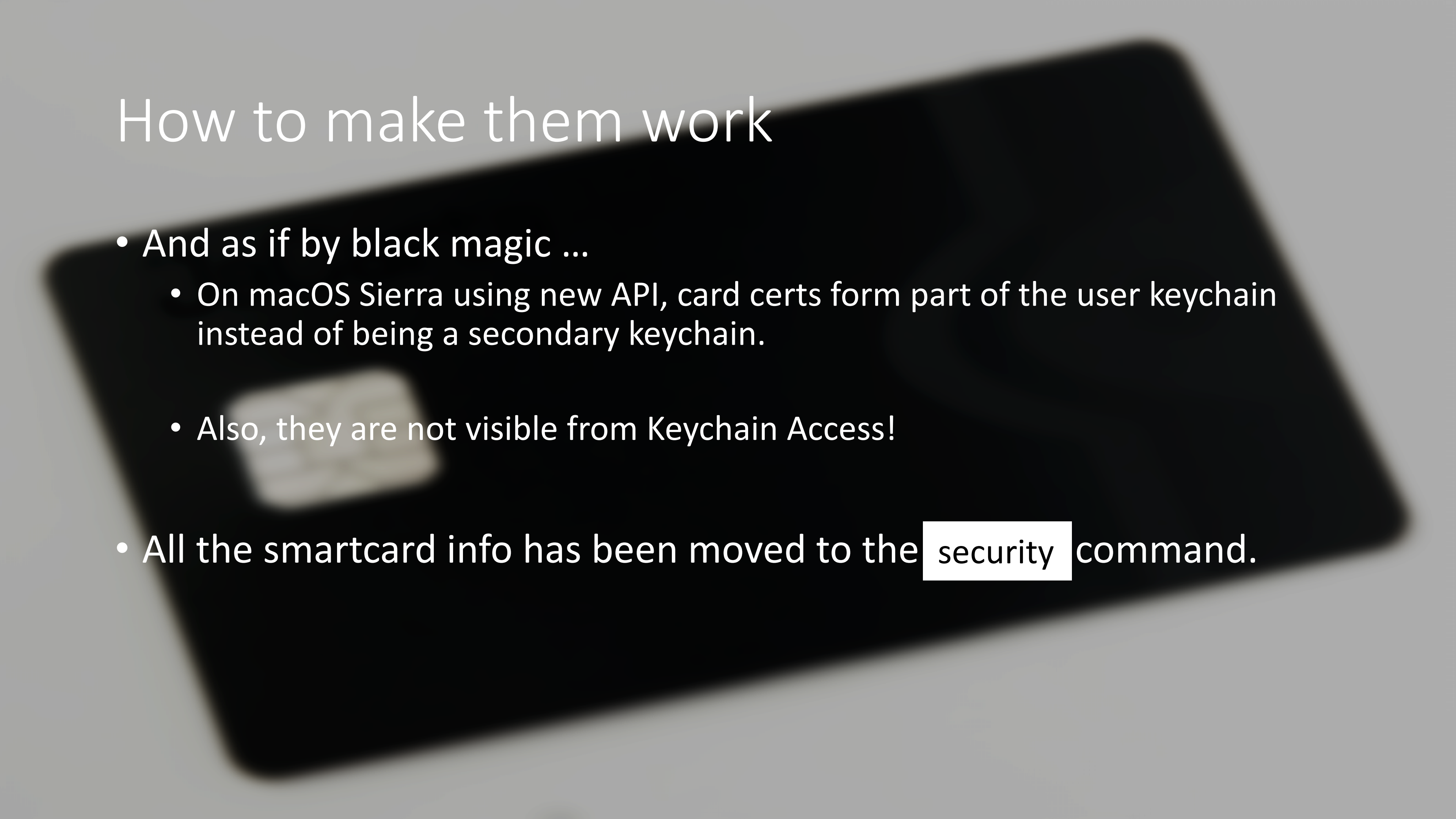


The Halcyon Days?
(10.12 onwards)



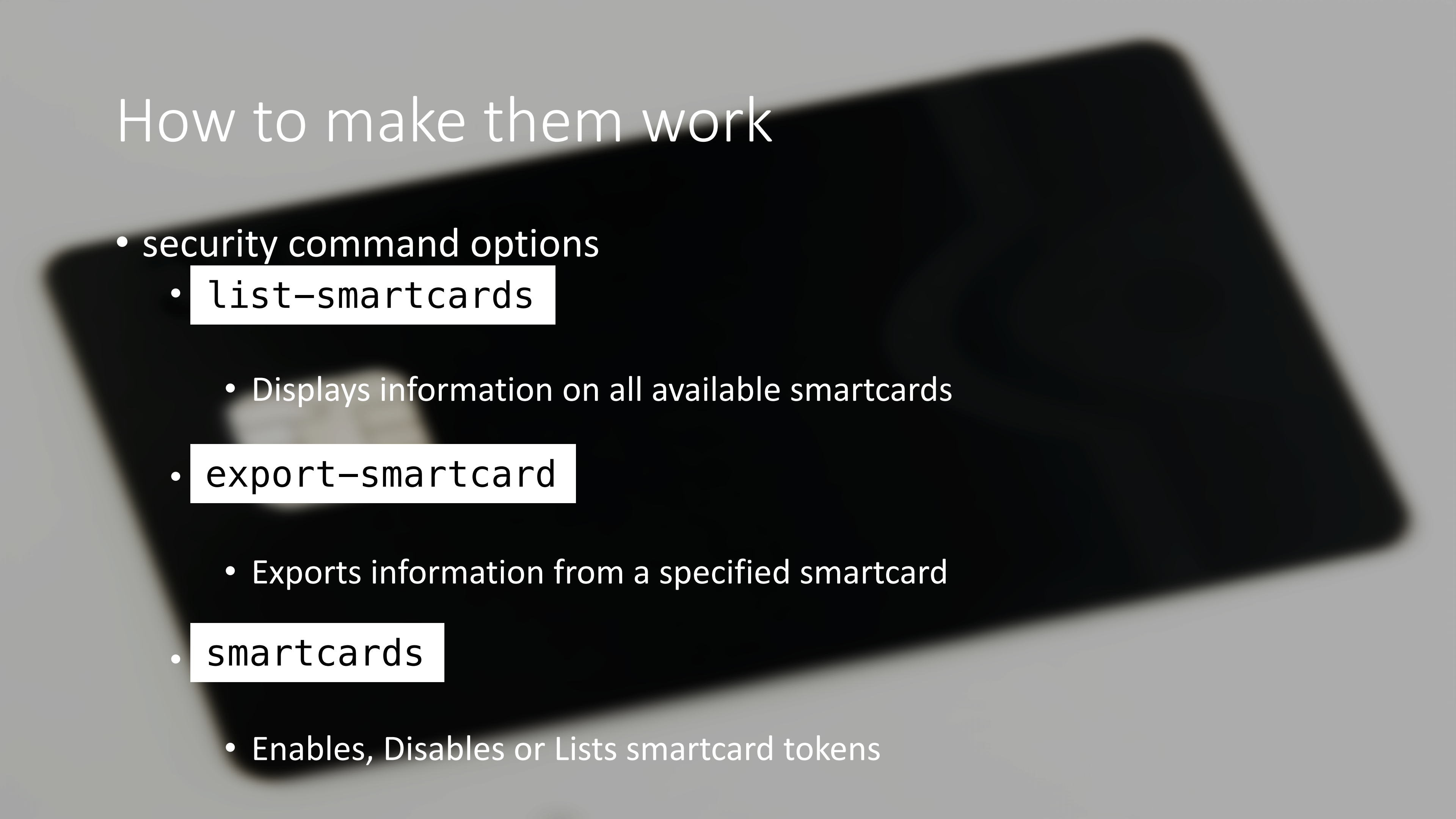
How to make them work

- macOS Configuration – Configuration Profile (10.12 CTK drivers)
  - PayloadType: com.apple.security.smartcard
  - Type: Boolean
  - Key: UserPairing
    - Enables/Disables the card pairing dialog, although existing pairings will still work.
  - Key: allowSmartCard
    - Enables/Disables the card for login, authorisations and screensaver unlocking.
  - Key: checkCertificateTrust
    - Enables/Disables certificate pinning for the card certificates.
  - Key: OneCardPerUser
    - Restricts a user to a single smart card. Doesn’t affect current cards parings.



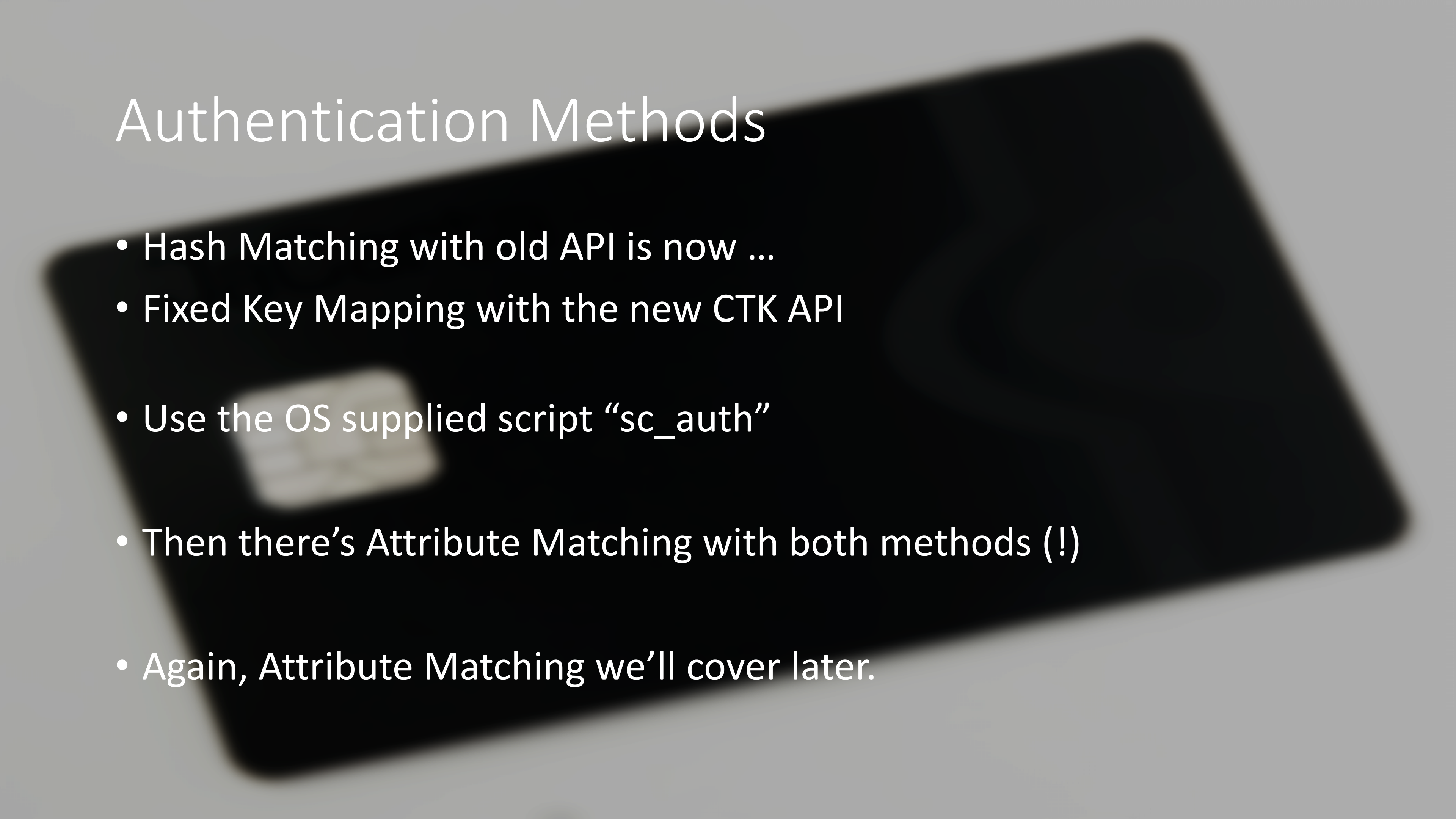
How to make them work

- And as if by black magic …
  - On macOS Sierra using new API, card certs form part of the user keychain instead of being a secondary keychain.
  - Also, they are not visible from Keychain Access!
- All the smartcard info has been moved to the `security` command.



How to make them work

- security command options
  - `list-smartcards`
    - Displays information on all available smartcards
  - `export-smartcard`
    - Exports information from a specified smartcard
  - `smartcards`
    - Enables, Disables or Lists smartcard tokens

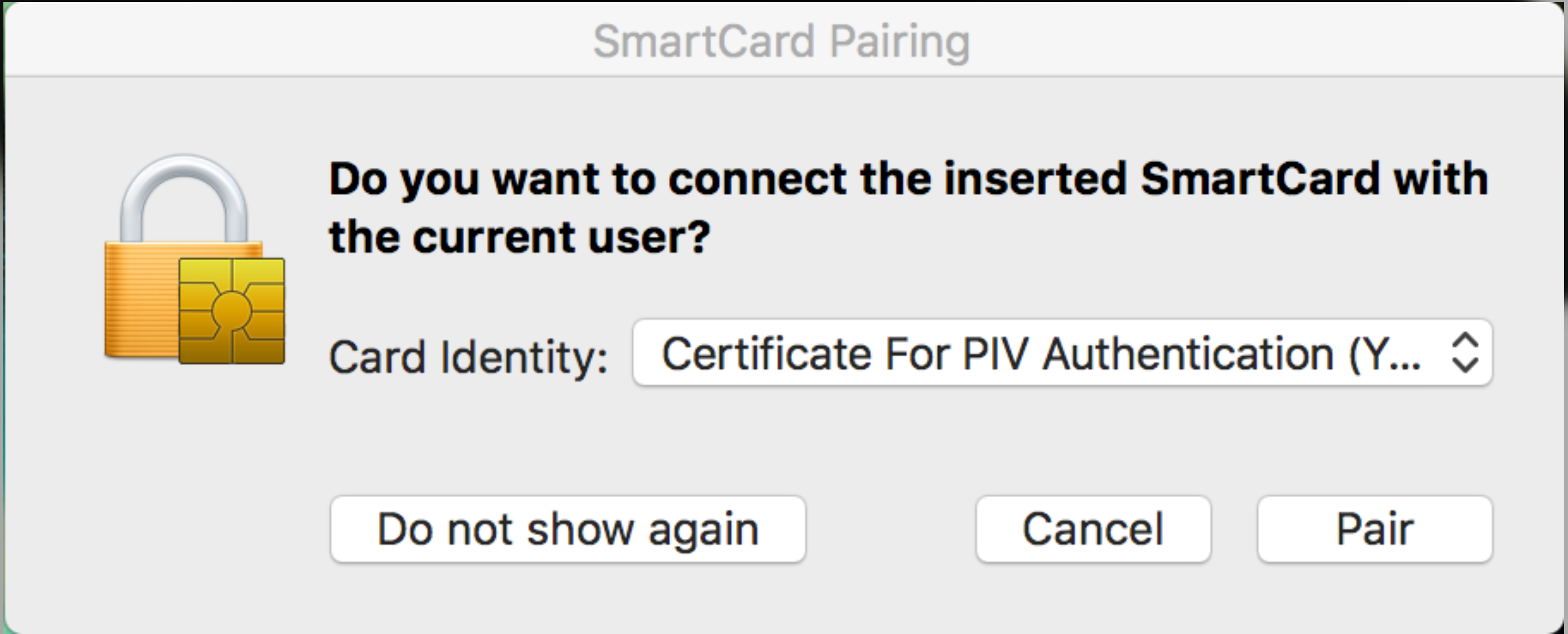


Authentication Methods

- Hash Matching with old API is now …
- Fixed Key Mapping with the new CTK API
- Use the OS supplied script “sc_auth”
- Then there’s Attribute Matching with both methods (!)
- Again, Attribute Matching we’ll cover later.

Authentication Methods

- Plug your CTK smart card in
  - and then …

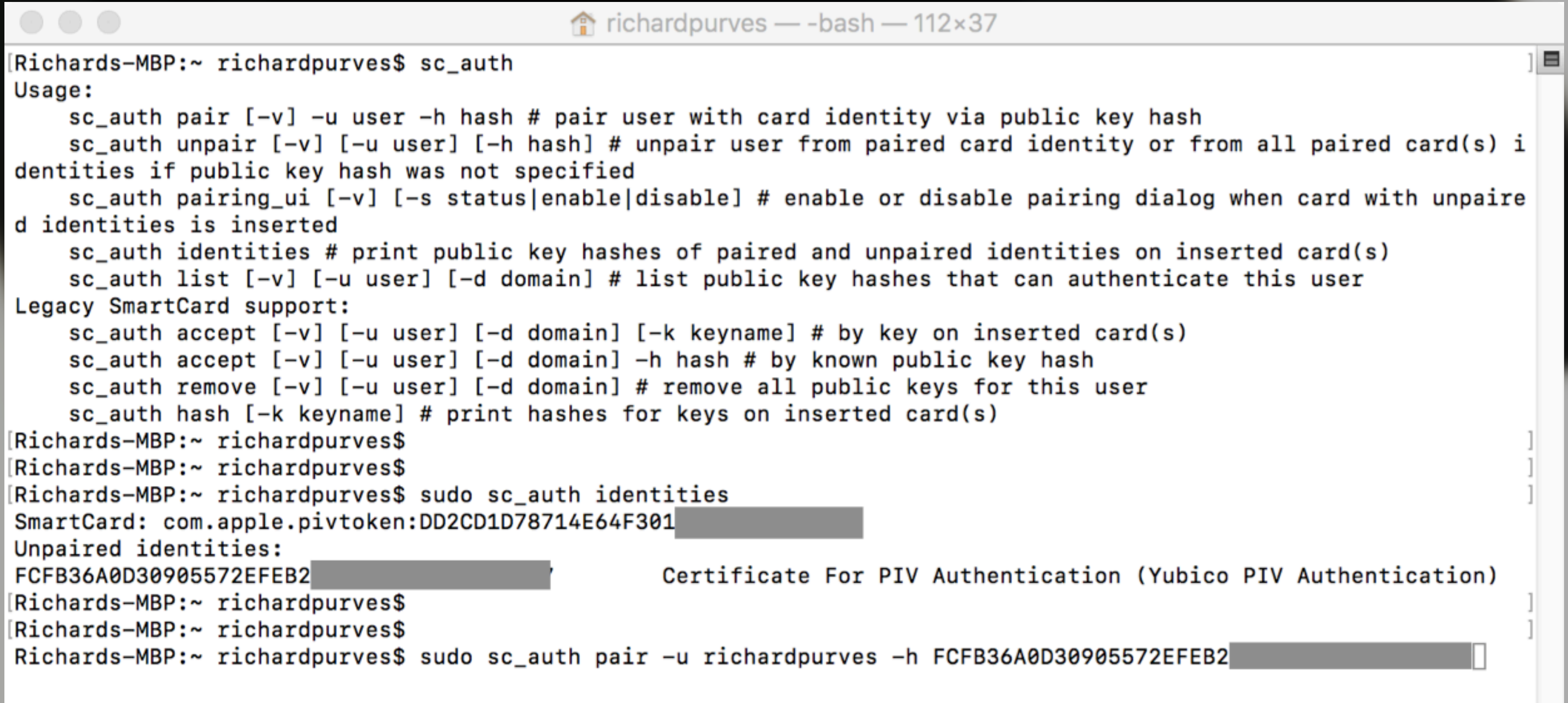

SmartCard Pairing

Do you want to connect the inserted SmartCard with the current user?

Card Identity: Certificate For PIV Authentication (Y...

Authentication Methods

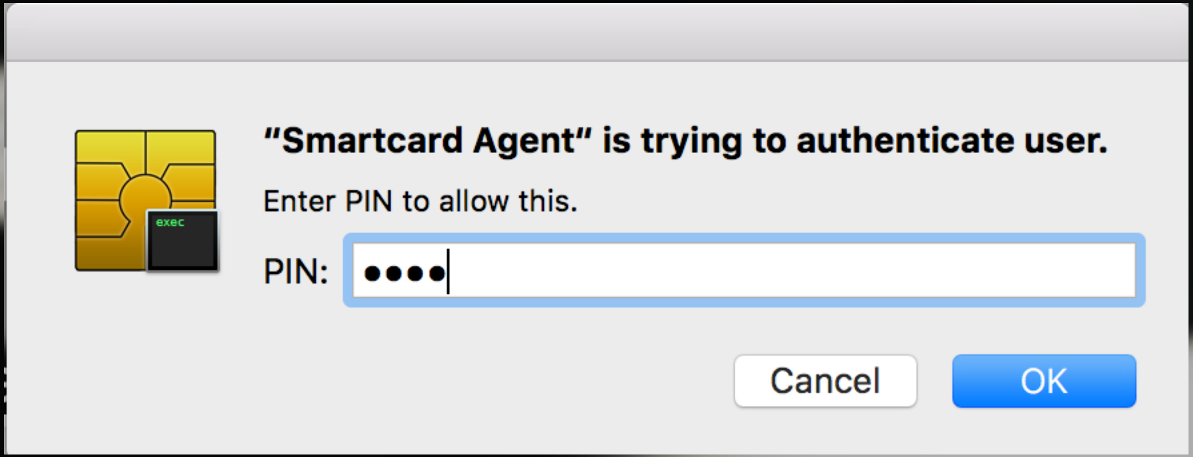
- Fixed Key Matching – sc_auth command


richardpurves — -bash — 112×37

```
[Richards-MBP:~ richardpurves$ sc_auth                                                                    ]
 Usage:
     sc_auth pair [-v] -u user -h hash # pair user with card identity via public key hash
     sc_auth unpair [-v] [-u user] [-h hash] # unpair user from paired card identity or from all paired card(s) i
 dentities if public key hash was not specified
     sc_auth pairing_ui [-v] [-s status|enable|disable] # enable or disable pairing dialog when card with unpaire
 d identities is inserted
     sc_auth identities # print public key hashes of paired and unpaired identities on inserted card(s)
     sc_auth list [-v] [-u user] [-d domain] # list public key hashes that can authenticate this user
 Legacy SmartCard support:
     sc_auth accept [-v] [-u user] [-d domain] [-k keyname] # by key on inserted card(s)
     sc_auth accept [-v] [-u user] [-d domain] -h hash # by known public key hash
     sc_auth remove [-v] [-u user] [-d domain] # remove all public keys for this user
     sc_auth hash [-k keyname] # print hashes for keys on inserted card(s)
[Richards-MBP:~ richardpurves$                                                                            ]
[Richards-MBP:~ richardpurves$                                                                            ]
[Richards-MBP:~ richardpurves$ sudo sc_auth identities                                                    ]
 SmartCard: com.apple.pivtoken:DD2CD1D78714E64F301[redacted]
 Unpaired identities:
 FCFB36A0D30905572EFEB2[redacted]        Certificate For PIV Authentication (Yubico PIV Authentication)
[Richards-MBP:~ richardpurves$                                                                            ]
[Richards-MBP:~ richardpurves$                                                                            ]
 Richards-MBP:~ richardpurves$ sudo sc_auth pair -u richardpurves -h FCFB36A0D30905572EFEB2[redacted]
```

Authentication Methods

- Fixed Key Matching – sc_auth command



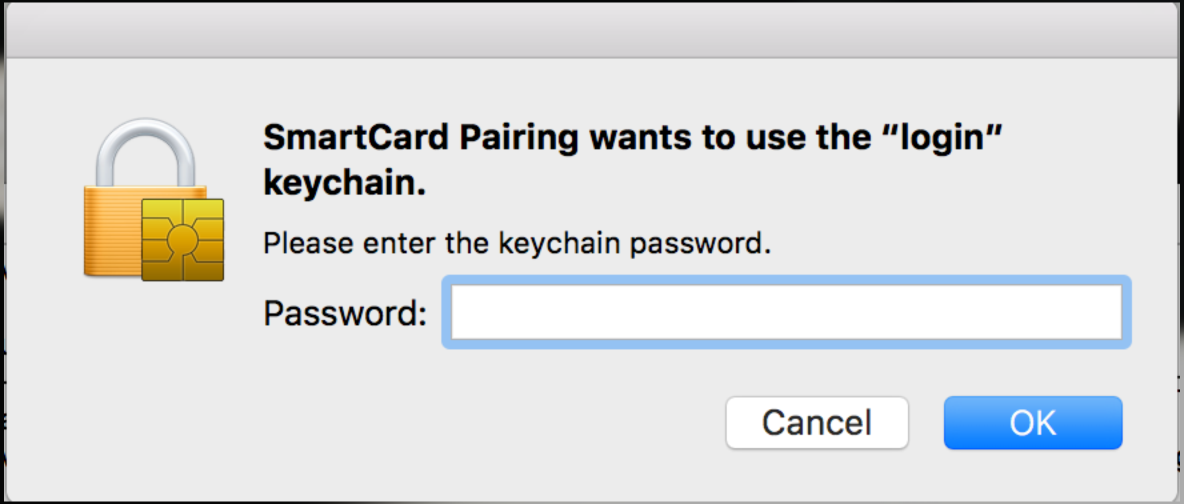
"Smartcard Agent" is trying to authenticate user.

Enter PIN to allow this.

PIN: ●●●●

Authentication Methods

- Fixed Key Matching – sc_auth command



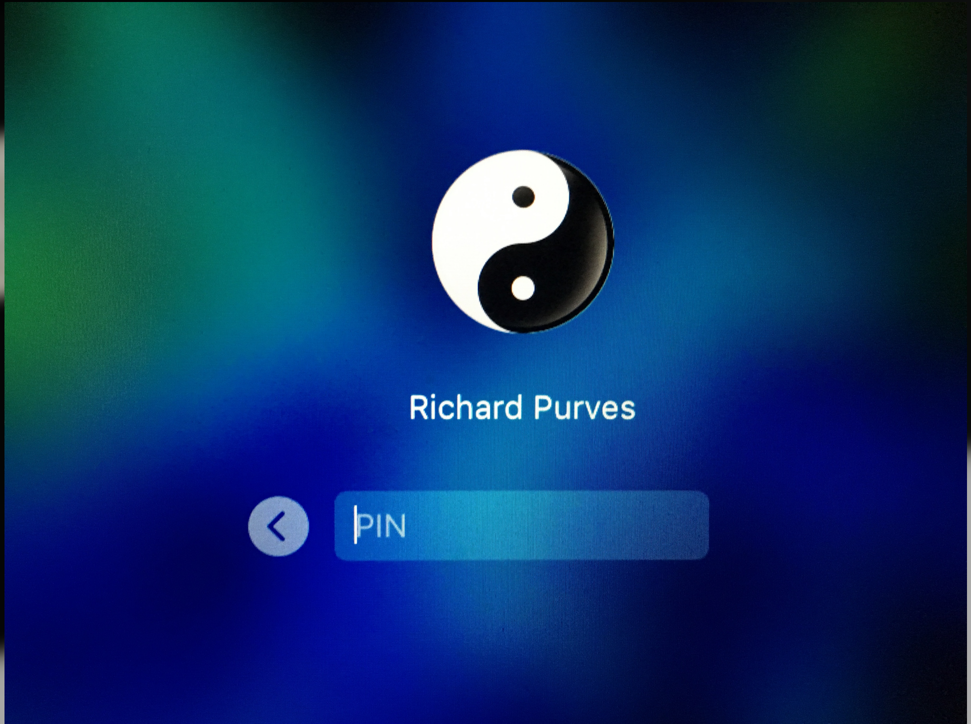
SmartCard Pairing wants to use the "login" keychain.

Please enter the keychain password.

Password:

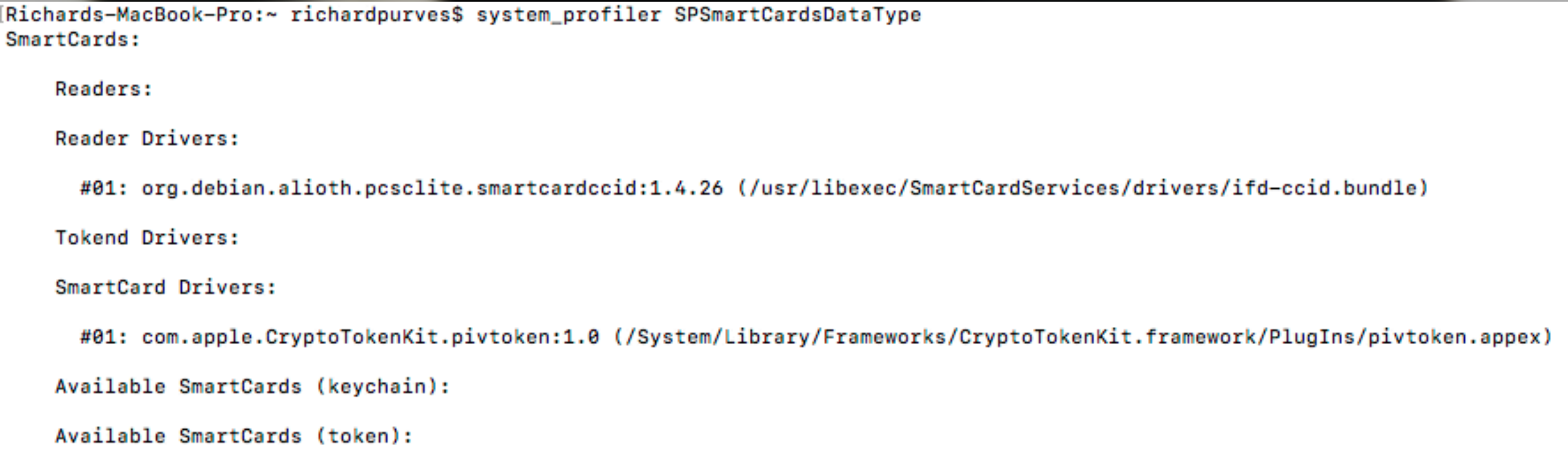
Authentication Methods

- Fixed Key Matching – sc_auth command

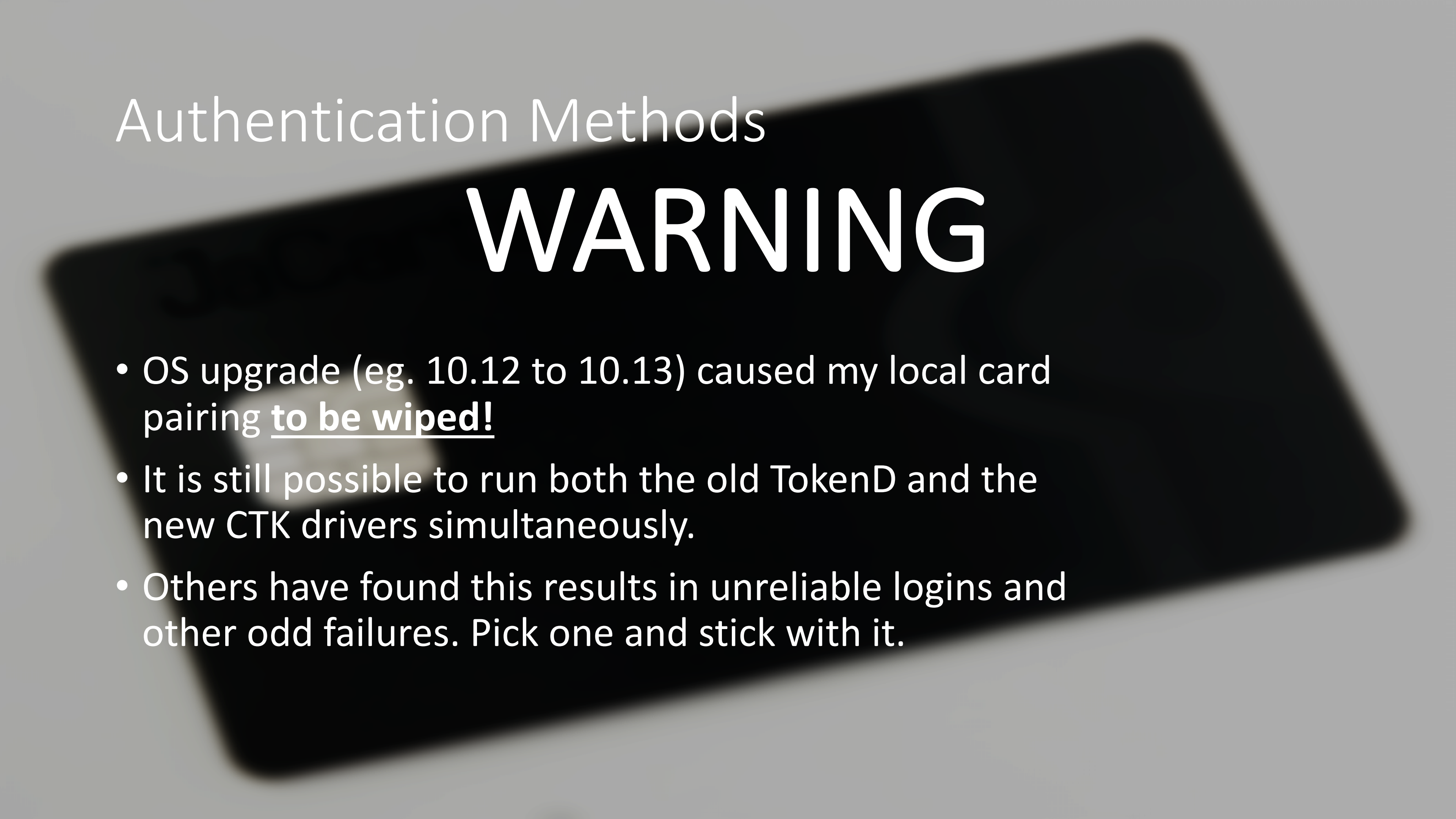


High Sierra Goodies!

- system_profiler extras in 10.13 !
  - `system_profiler SPSmartCardsDataType`



```
Richards-MacBook-Pro:~ richardpurves$ system_profiler SPSmartCardsDataType
SmartCards:

    Readers:

    Reader Drivers:

      #01: org.debian.alioth.pcsclite.smartcardccid:1.4.26 (/usr/libexec/SmartCardServices/drivers/ifd-ccid.bundle)

    Tokend Drivers:

    SmartCard Drivers:

      #01: com.apple.CryptoTokenKit.pivtoken:1.0 (/System/Library/Frameworks/CryptoTokenKit.framework/PlugIns/pivtoken.appex)

    Available SmartCards (keychain):

    Available SmartCards (token):
```



Authentication Methods

WARNING

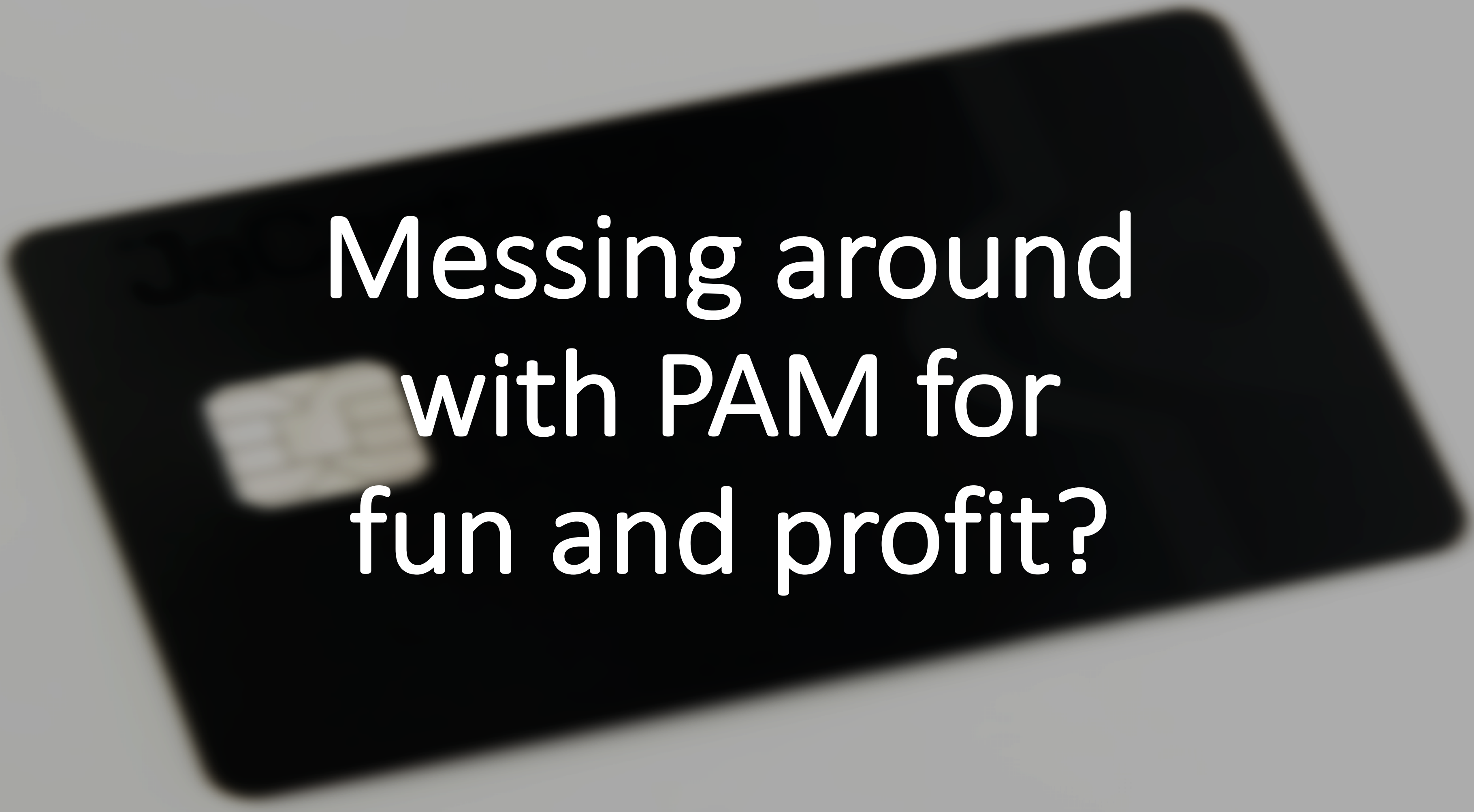
- OS upgrade (eg. 10.12 to 10.13) caused my local card pairing **to be wiped!**
- It is still possible to run both the old TokenD and the new CTK drivers simultaneously.
- Others have found this results in unreliable logins and other odd failures. Pick one and stick with it.

Authentication Methods

- Disabling CTK plugins

PayloadType:	com.apple.security.smartcard
Type:	array
Key:	DisabledTokens
Array:	com.apple.CryptoTokenKit.pivtoken

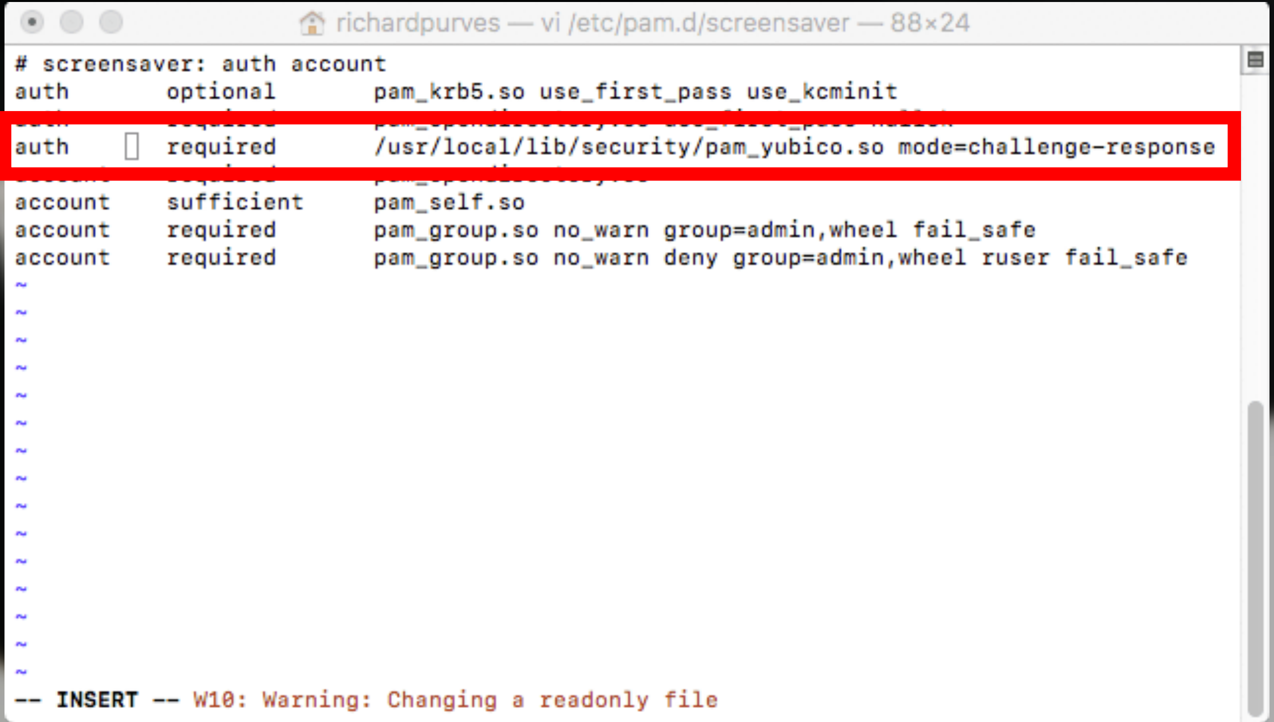
- Replace “pivtoken” with card format(s) of your choice



Messing around with PAM for fun and profit?

PAM Modification

- Screensaver locking - /etc/pam.d/screensaver

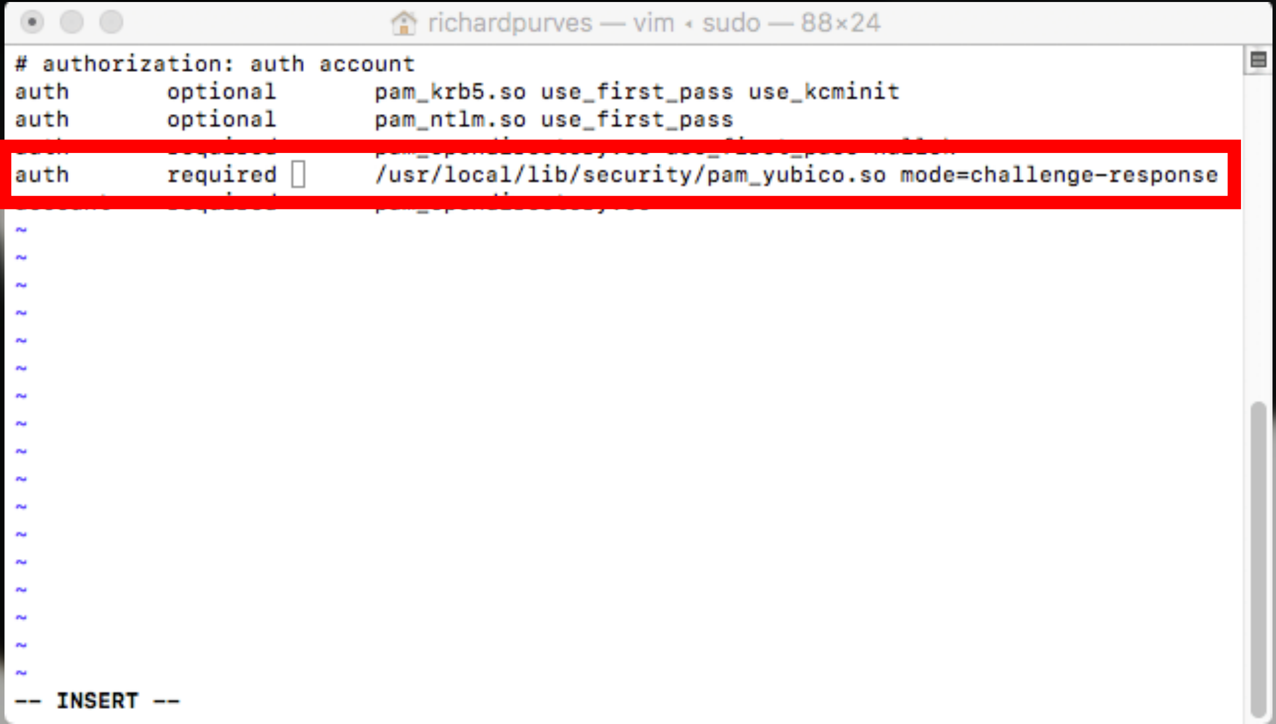


richardpurves — vi /etc/pam.d/screensaver — 88×24

```
# screensaver: auth account
auth        optional        pam_krb5.so use_first_pass use_kcminit
[illegible]
auth        required        /usr/local/lib/security/pam_yubico.so mode=challenge-response
[illegible]
account     sufficient      pam_self.so
account     required        pam_group.so no_warn group=admin,wheel fail_safe
account     required        pam_group.so no_warn deny group=admin,wheel ruser fail_safe
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT -- W10: Warning: Changing a readonly file
```

PAM Modification

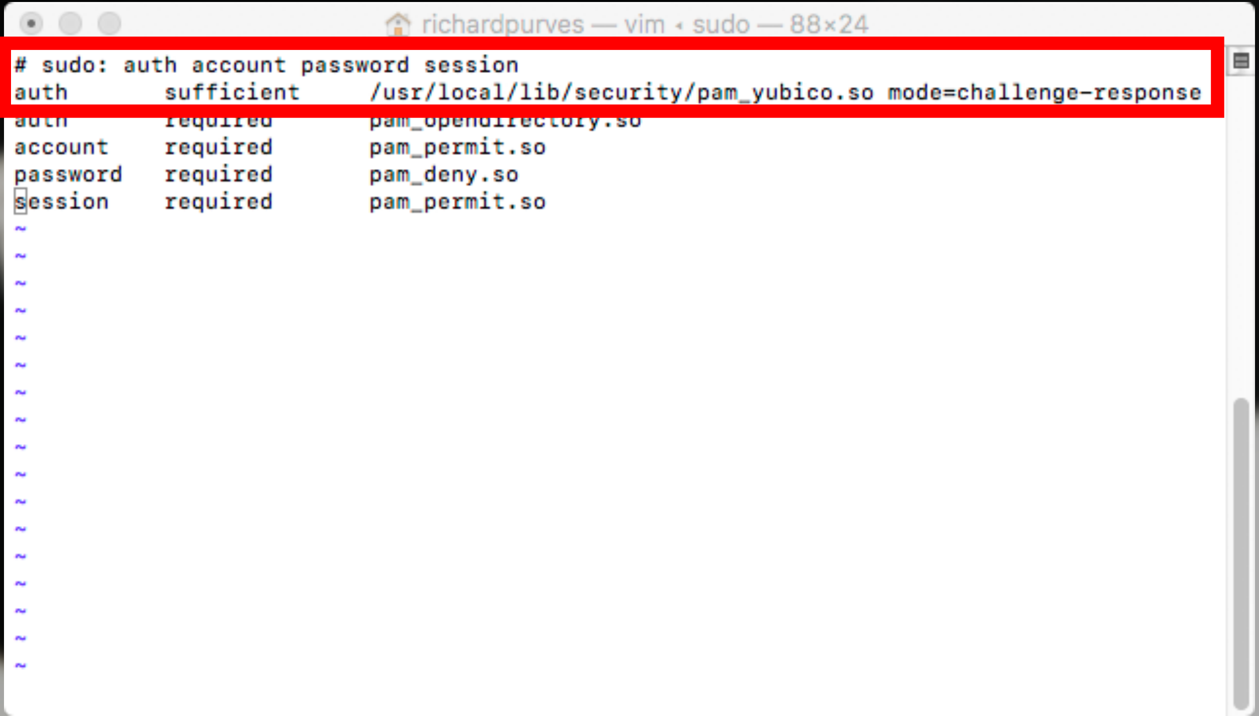
- Loginwindow locking - /etc/pam.d/authorization



```
# authorization: auth account
auth       optional       pam_krb5.so use_first_pass use_kcminit
auth       optional       pam_ntlm.so use_first_pass
auth       required       pam_opendirectory.so use_first_pass nullok
auth       required       /usr/local/lib/security/pam_yubico.so mode=challenge-response
account    required       pam_opendirectory.so
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --
```

PAM Modification

- sudo locking - /etc/pam.d/sudo

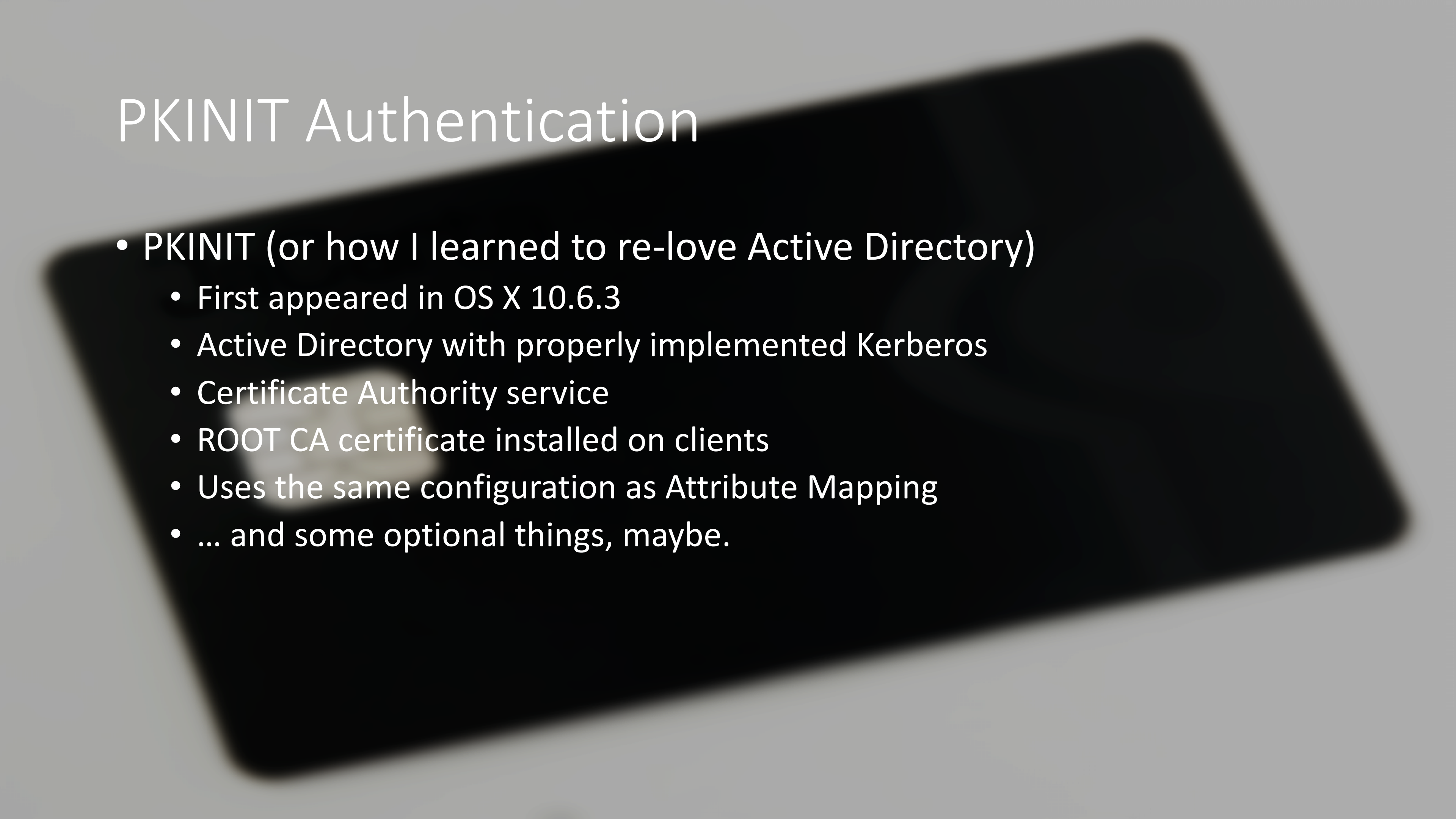


```
# sudo: auth account password session
auth       sufficient     /usr/local/lib/security/pam_yubico.so mode=challenge-response
auth       required       pam_opendirectory.so
account    required       pam_permit.so
password   required       pam_deny.so
session    required       pam_permit.so
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

PKINIT (or how I ~~sort of~~ learned to re-love Active Directory)

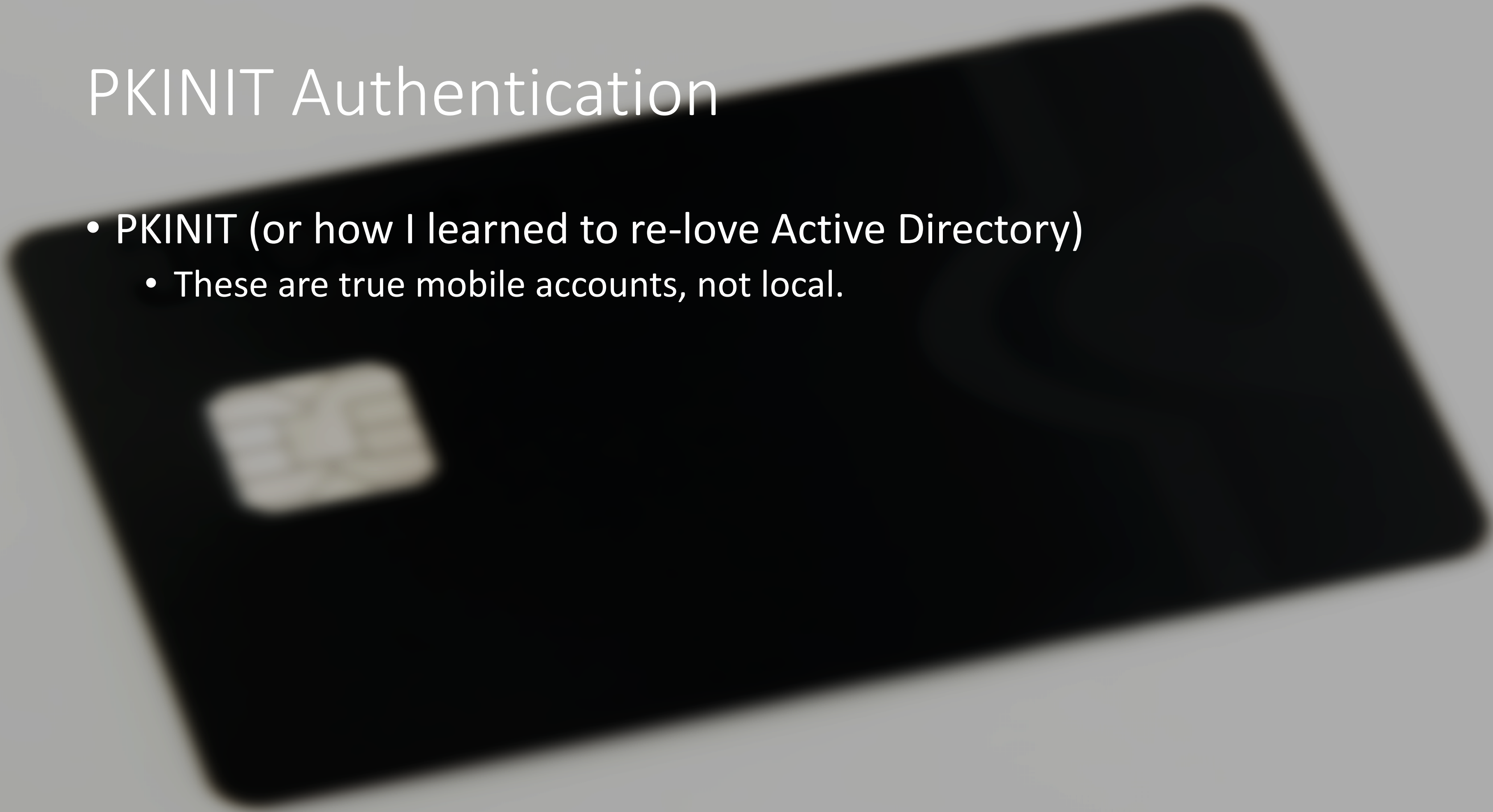
PKINIT Authentication

- PKINIT (or how I learned to re-love Active Directory)
  - First appeared in OS X 10.6.3
  - Active Directory with properly implemented Kerberos
  - Certificate Authority service
  - ROOT CA certificate installed on clients
  - Uses the same configuration as Attribute Mapping
  - … and some optional things, maybe.



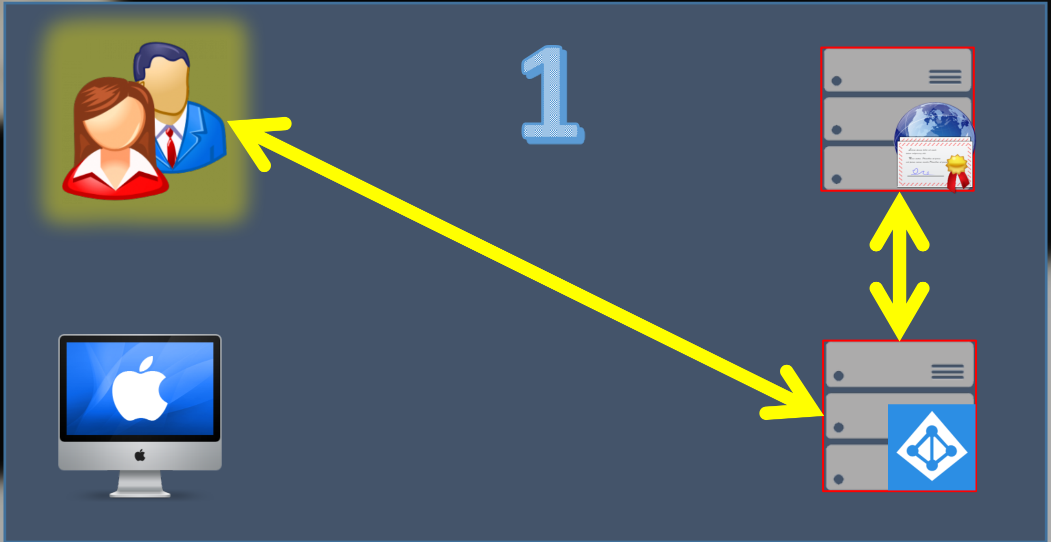
PKINIT Authentication

- PKINIT (or how I learned to re-love Active Directory)
  - These are true mobile accounts, not local.



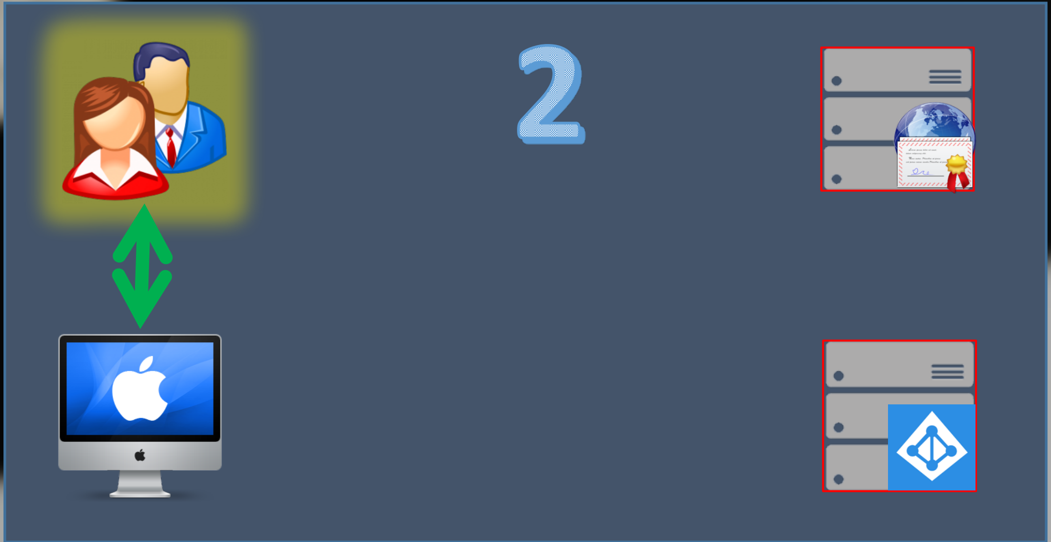
PKINIT Authentication

- PKINIT – The (simplified) Process
  - User account is provisioned with an individual certificate from the CA.



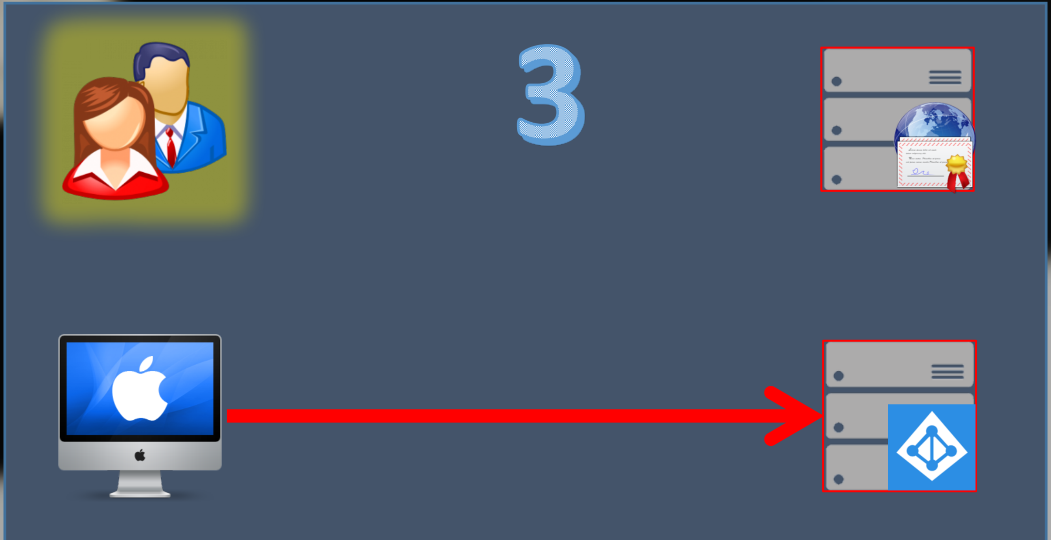
PKINIT Authentication

- PKINIT – The (simplified) Process
  - User unlocks the cert on the smart card, allowing the OS to use it.



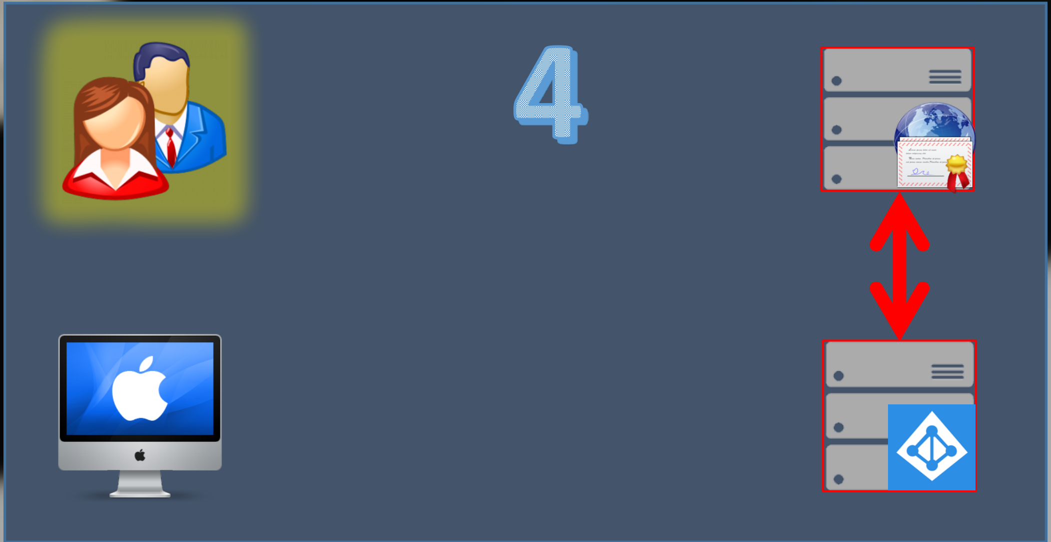
PKINIT Authentication

- PKINIT – The (simplified) Process
  - OS sends a TGT Request to the Active Directory DC(s) using certificate from the smart card



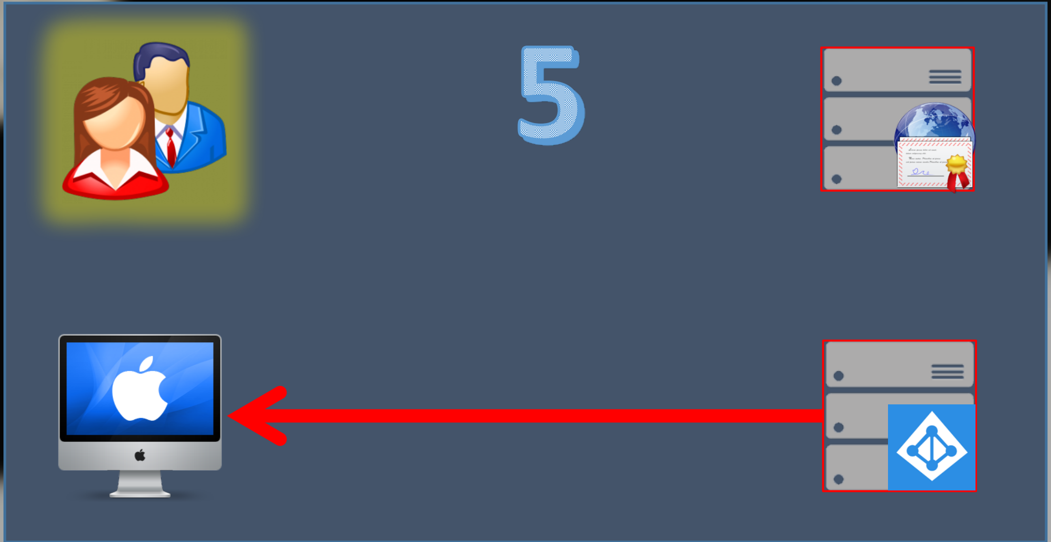
PKINIT Authentication

- PKINIT – The (simplified) Process
  - DC checks the supplied cert against the record in the Certificate Authority



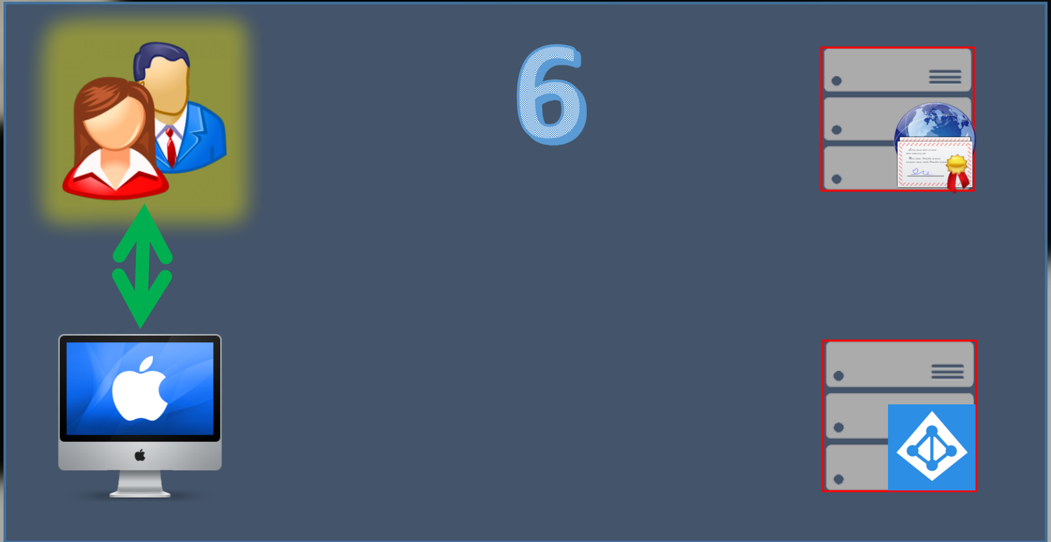
PKINIT Authentication

- PKINIT – The (simplified) Process
  - TGT Reply sent back from the DC to the Mac client.
  - (either a Kerberos Ticket or a denial)



PKINIT Authentication

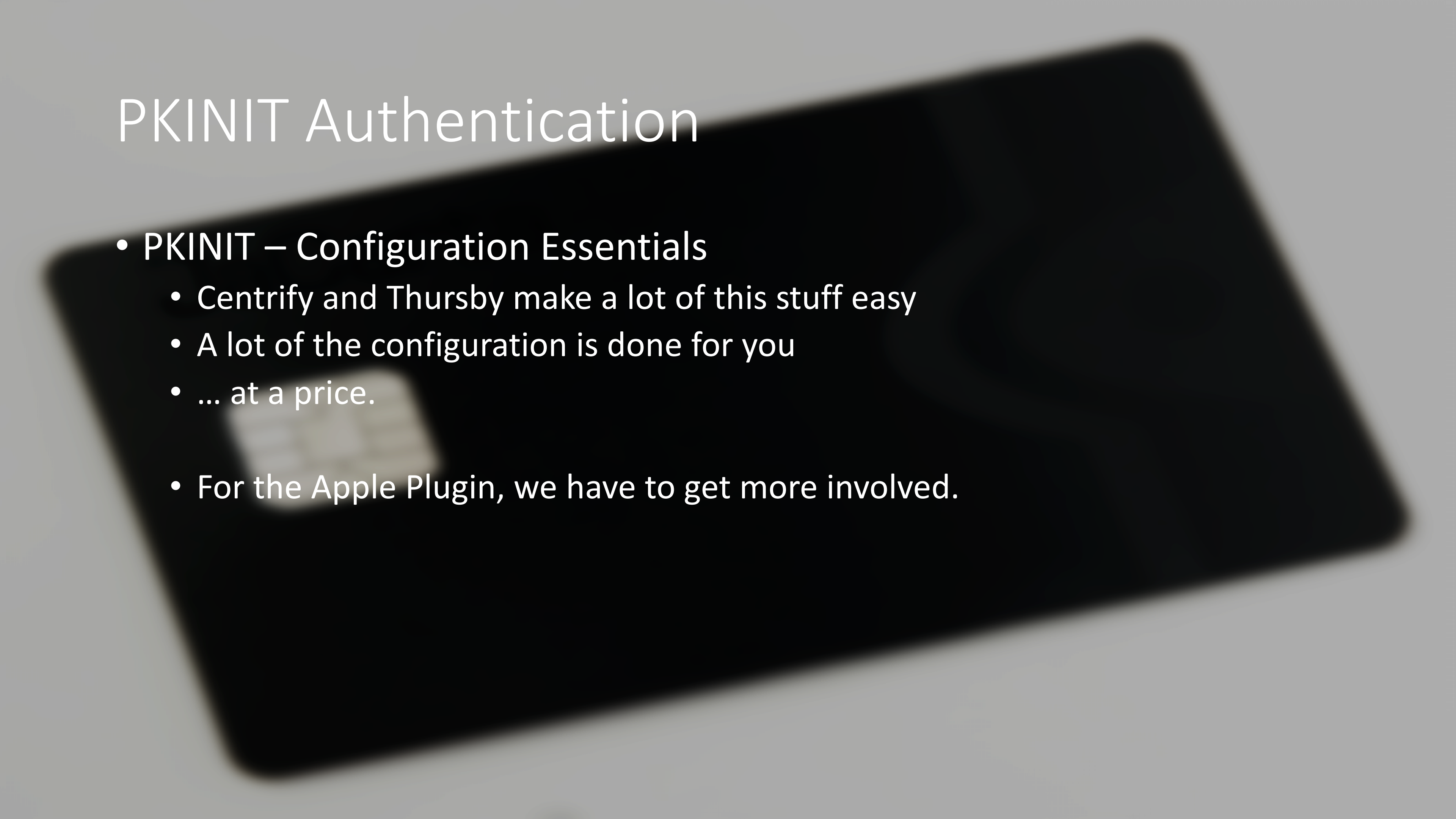
- PKINIT – The (simplified) Process
  - Unique in that the user does NOT have a password on their AD account!



PKINIT Authentication

- PKINIT – Configuration Essentials
  - AD Bind (Apple Plugin, Centrify or AdmitMac PKI)
  - The stuff from ~~scene 24~~ slides 49, 52 and 61.
  - Disable FileVault 2 automatic user login (more on this soon)

• PayloadType:	com.apple.loginwindow
• Key:	DisableFDEAutoLogin
• Type:	boolean
• Setting:	YES



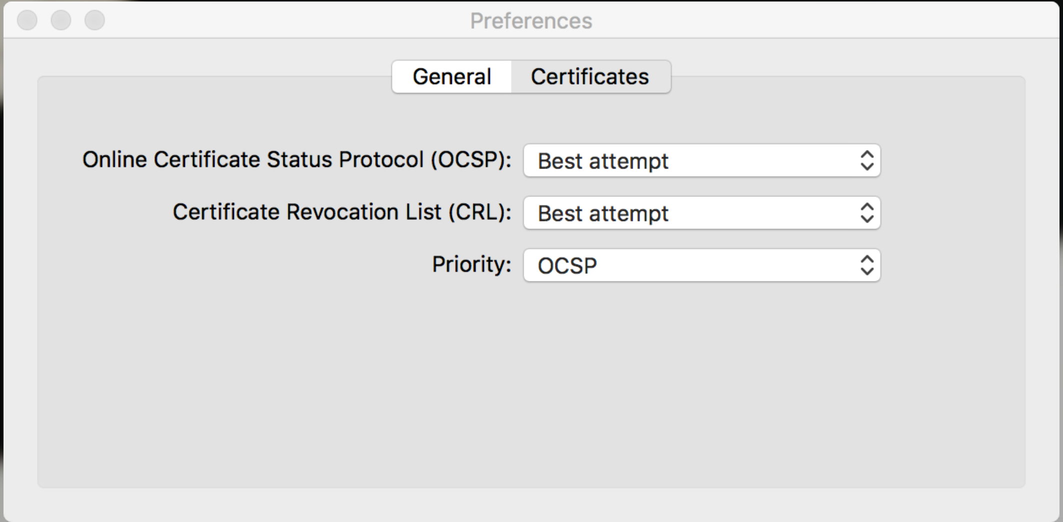
PKINIT Authentication

- PKINIT – Configuration Essentials
  - Centrify and Thursby make a lot of this stuff easy
  - A lot of the configuration is done for you
  - … at a price.

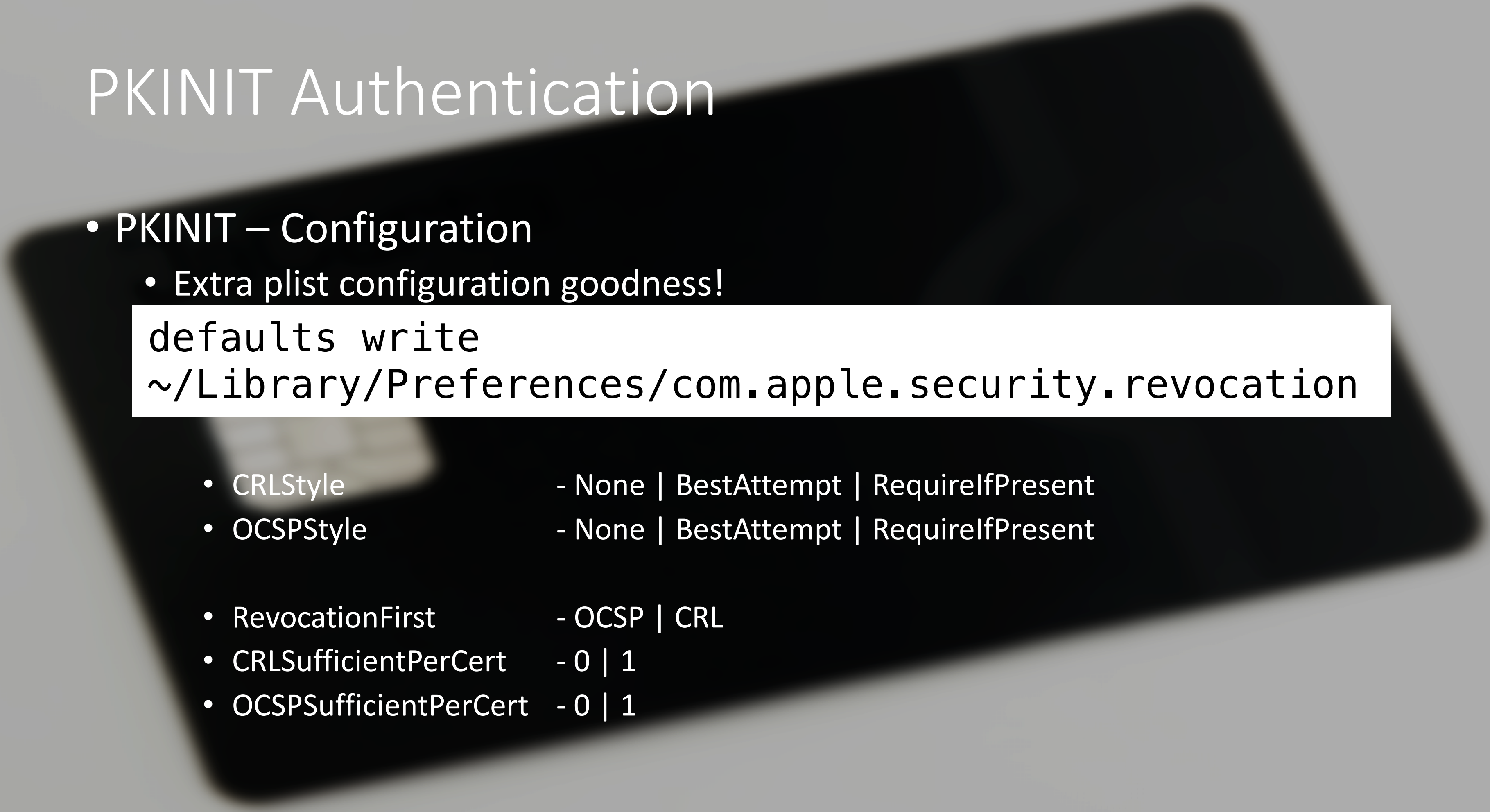
  - For the Apple Plugin, we have to get more involved.

PKINIT Authentication

- PKINIT – Configuration
  - Extra plist configuration goodness!
  - Keychain Access app has certificate security settings



Online Certificate Status Protocol (OCSP): Best attempt
Certificate Revocation List (CRL): Best attempt
Priority: OCSP

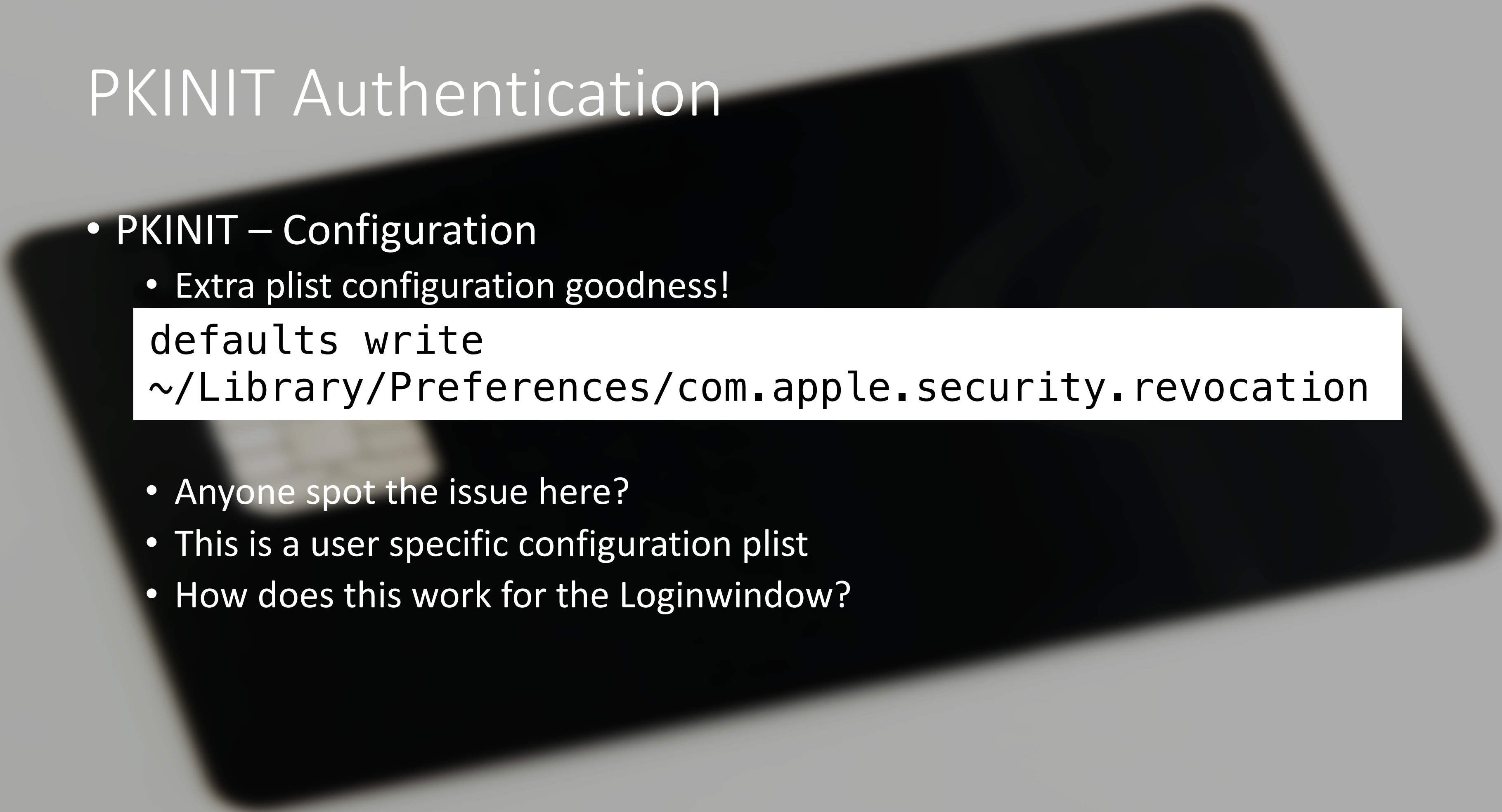


PKINIT Authentication

- PKINIT – Configuration
  - Extra plist configuration goodness!

```
defaults write
~/Library/Preferences/com.apple.security.revocation
```

CRLStyle	- None | BestAttempt | RequireIfPresent
OCSPStyle	- None | BestAttempt | RequireIfPresent
RevocationFirst	- OCSP | CRL
CRLSufficientPerCert	- 0 | 1
OCSPSufficientPerCert	- 0 | 1



PKINIT Authentication

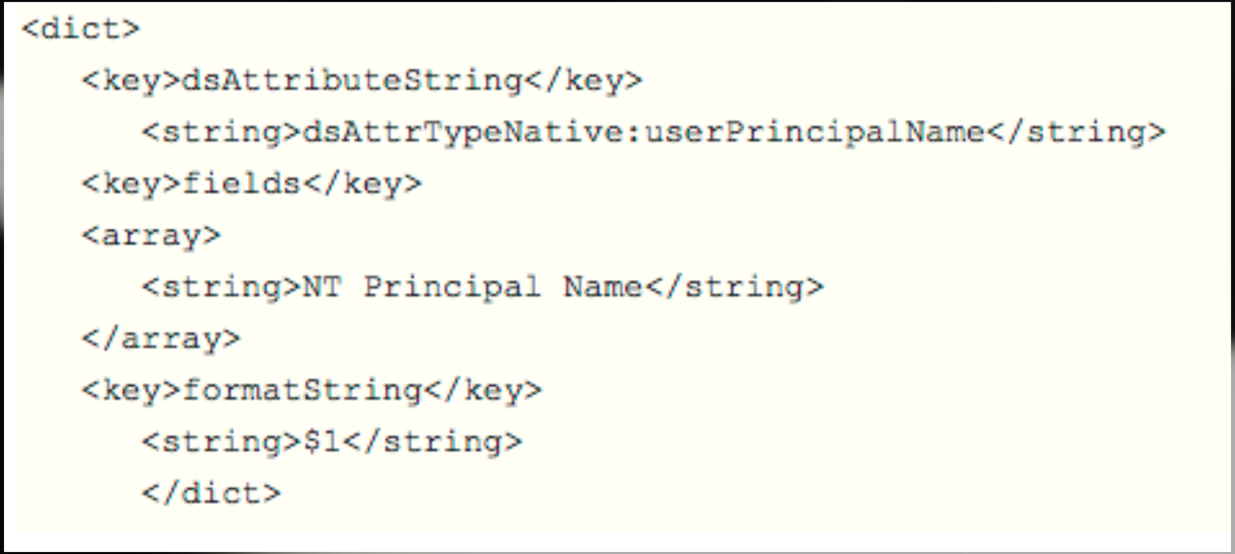
- PKINIT – Configuration
  - Extra plist configuration goodness!

```
defaults write
~/Library/Preferences/com.apple.security.revocation
```

  - Anyone spot the issue here?
  - This is a user specific configuration plist
  - How does this work for the Loginwindow?

PKINIT Authentication

- PKINIT – 10.6.3 onward with legacy drivers
  - /etc/cacloginconfig.plist

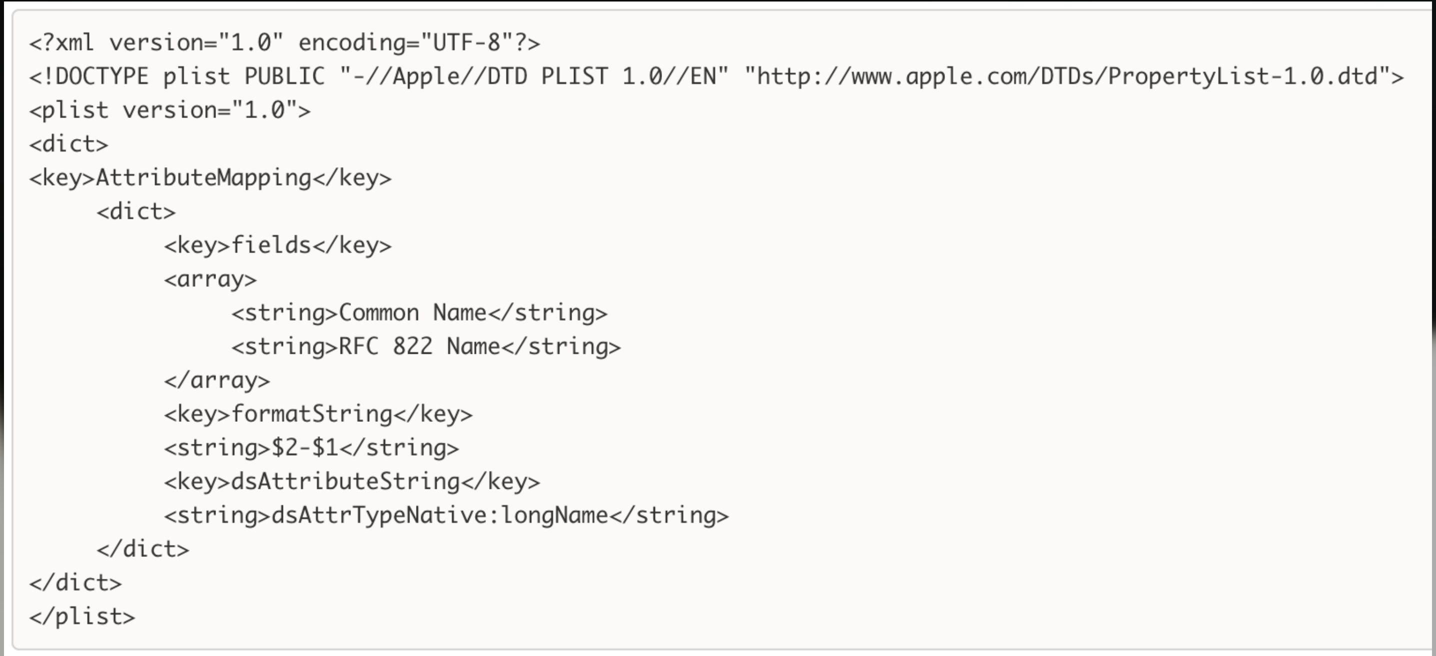


```
<dict>
   <key>dsAttributeString</key>
      <string>dsAttrTypeNative:userPrincipalName</string>
   <key>fields</key>
   <array>
      <string>NT Principal Name</string>
   </array>
   <key>formatString</key>
      <string>$1</string>
      </dict>
```

  - This same file is also used for “Attribute Matching”.

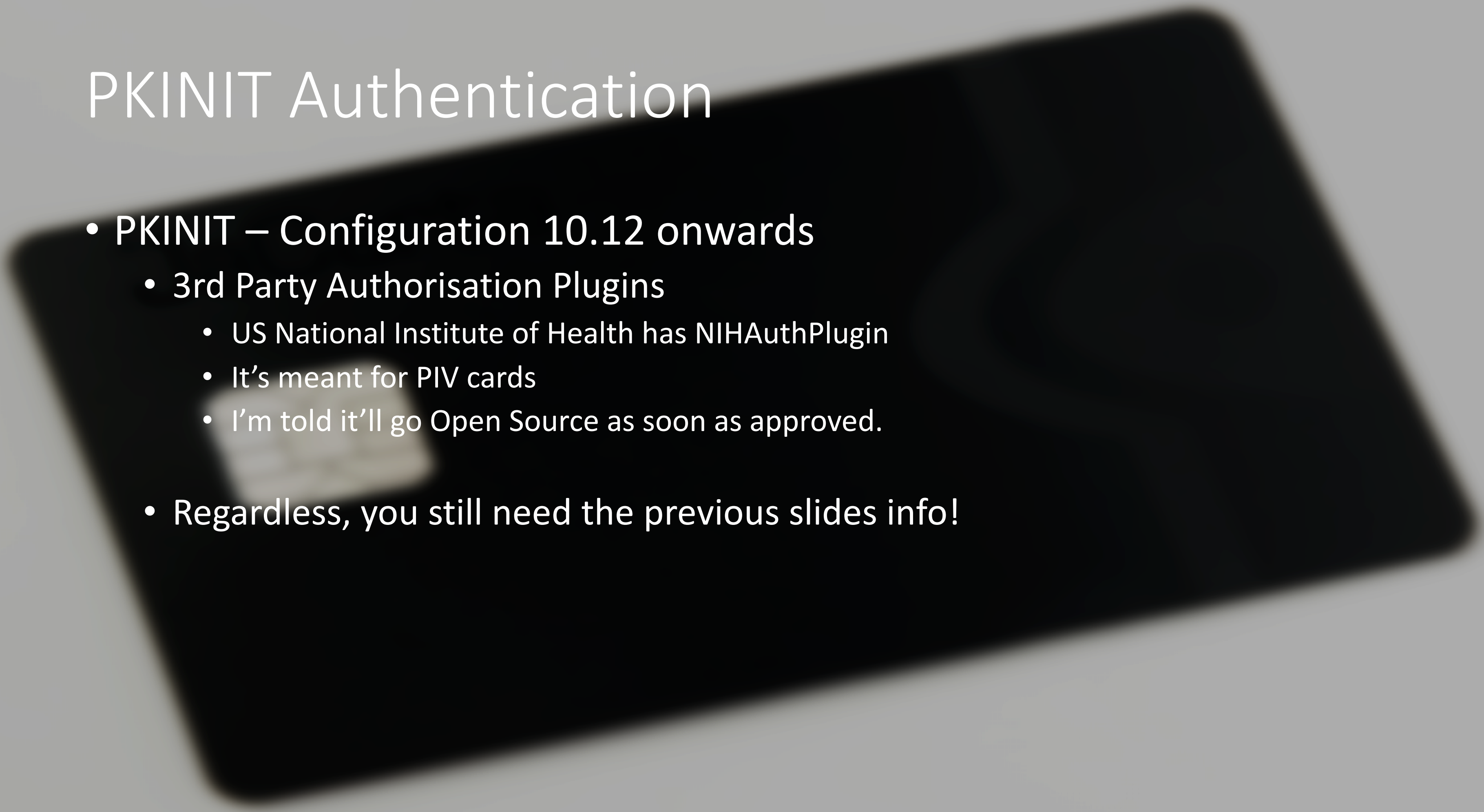
PKINIT Authentication

- PKINIT – 10.12 onwards for CTK API drivers
    - /etc/SmartcardConfig.plist

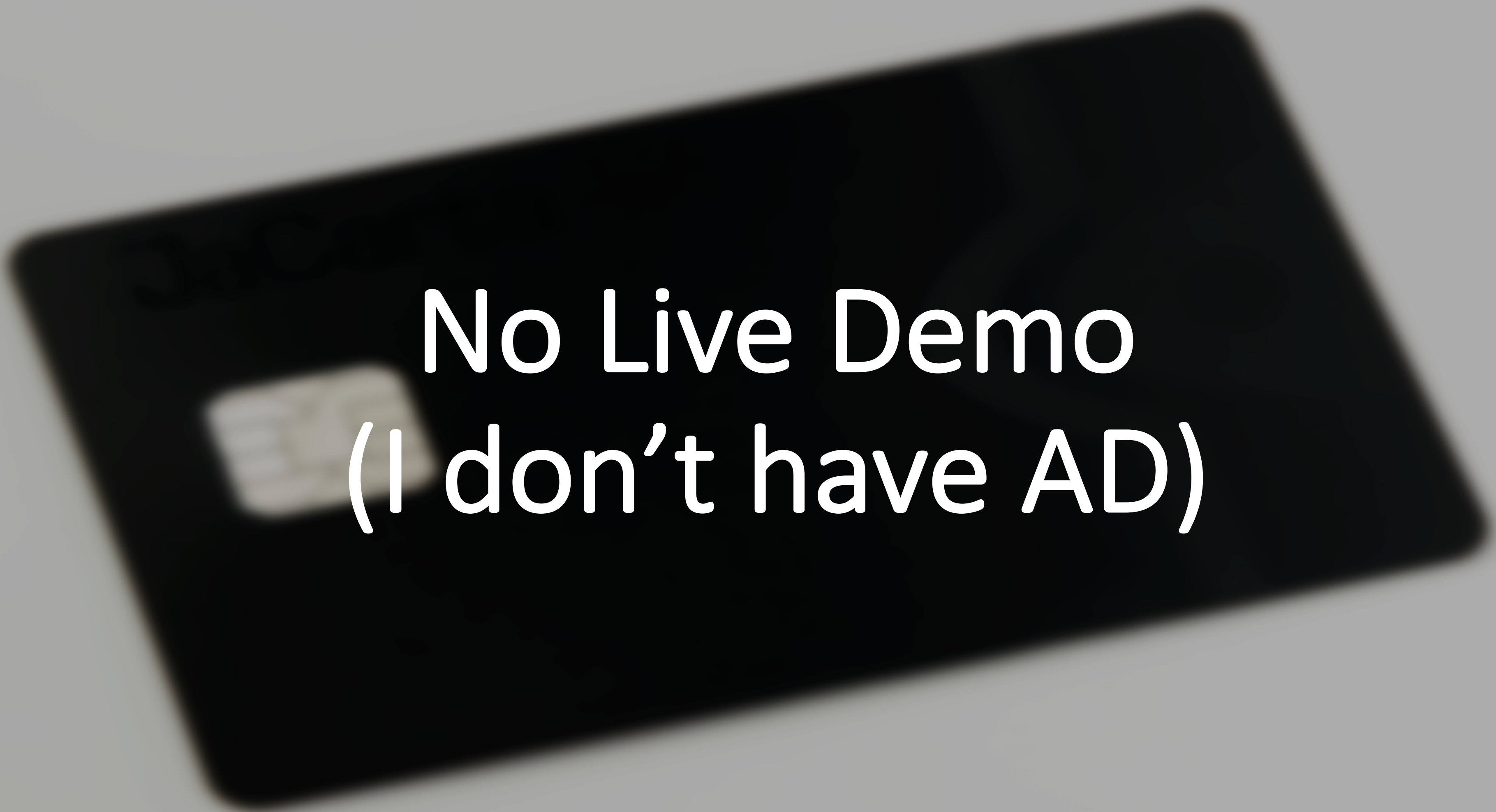


```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
<key>AttributeMapping</key>
     <dict>
          <key>fields</key>
          <array>
               <string>Common Name</string>
               <string>RFC 822 Name</string>
          </array>
          <key>formatString</key>
          <string>$2-$1</string>
          <key>dsAttributeString</key>
          <string>dsAttrTypeNative:longName</string>
     </dict>
</dict>
</plist>
```

PKINIT Authentication



- PKINIT – Configuration 10.12 onwards
  - 3rd Party Authorisation Plugins
    - US National Institute of Health has NIHAuthPlugin
    - It’s meant for PIV cards
    - I’m told it’ll go Open Source as soon as approved.
  - Regardless, you still need the previous slides info!

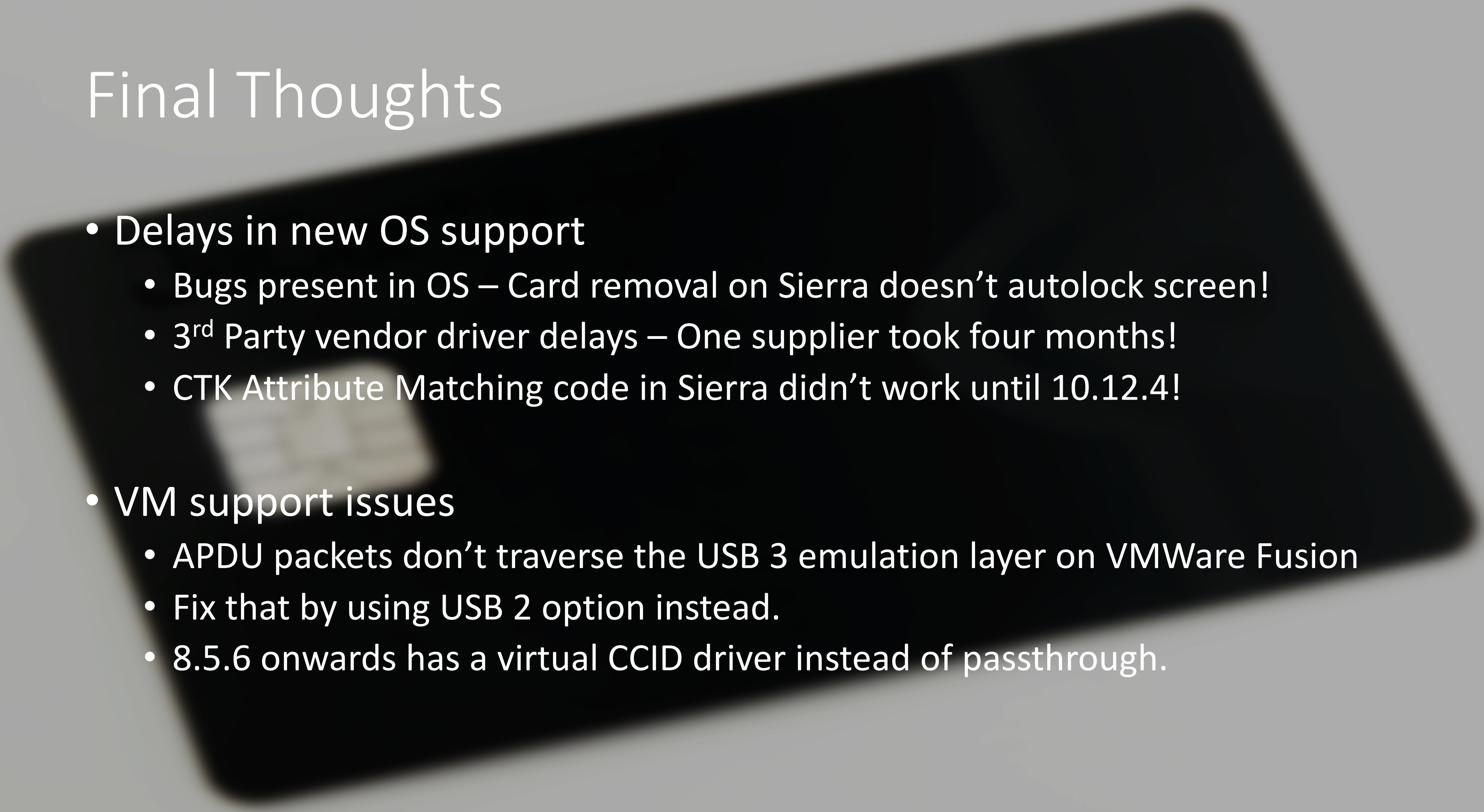


No Live Demo
(I don’t have AD)

Or you could just use NoMAD …

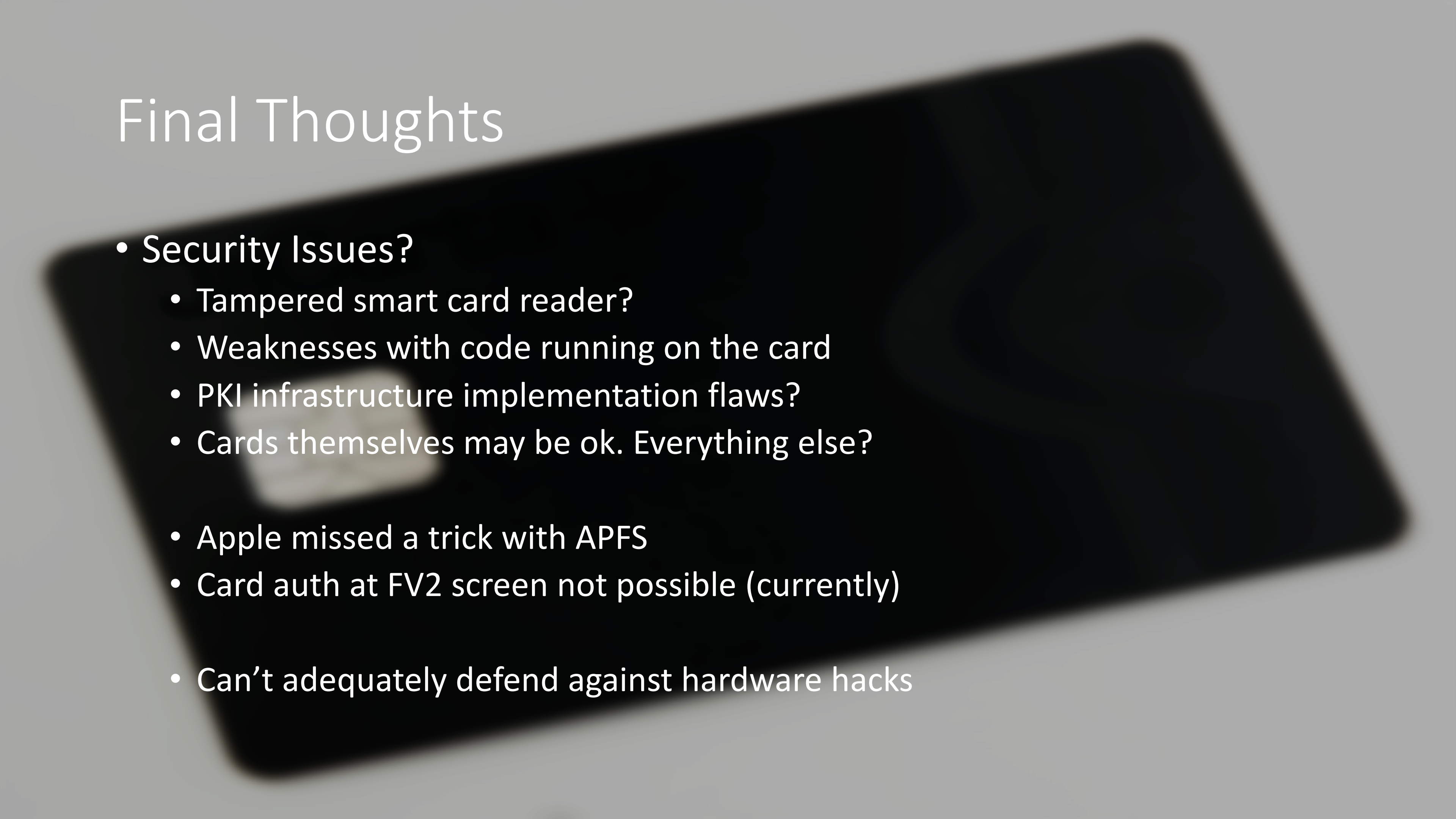
Or you could just pay for Enterprise Connect PKI

~~Jerry Springer's~~ Final Thoughts



Final Thoughts

- Delays in new OS support
  - Bugs present in OS – Card removal on Sierra doesn’t autolock screen!
  - 3rd Party vendor driver delays – One supplier took four months!
  - CTK Attribute Matching code in Sierra didn’t work until 10.12.4!
- VM support issues
  - APDU packets don’t traverse the USB 3 emulation layer on VMWare Fusion
  - Fix that by using USB 2 option instead.
  - 8.5.6 onwards has a virtual CCID driver instead of passthrough.

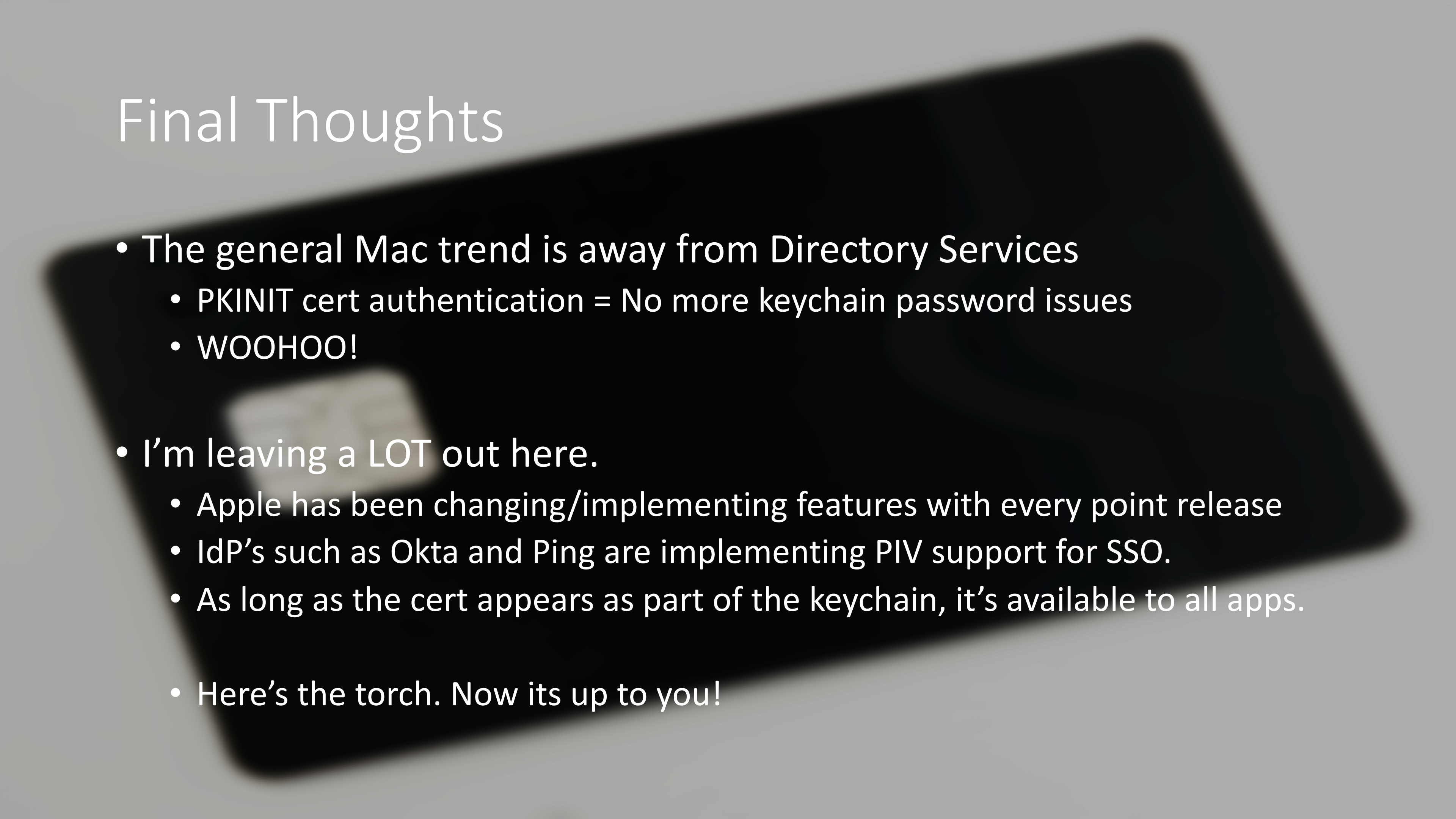


Final Thoughts

- Security Issues?
  - Tampered smart card reader?
  - Weaknesses with code running on the card
  - PKI infrastructure implementation flaws?
  - Cards themselves may be ok. Everything else?

  - Apple missed a trick with APFS
  - Card auth at FV2 screen not possible (currently)

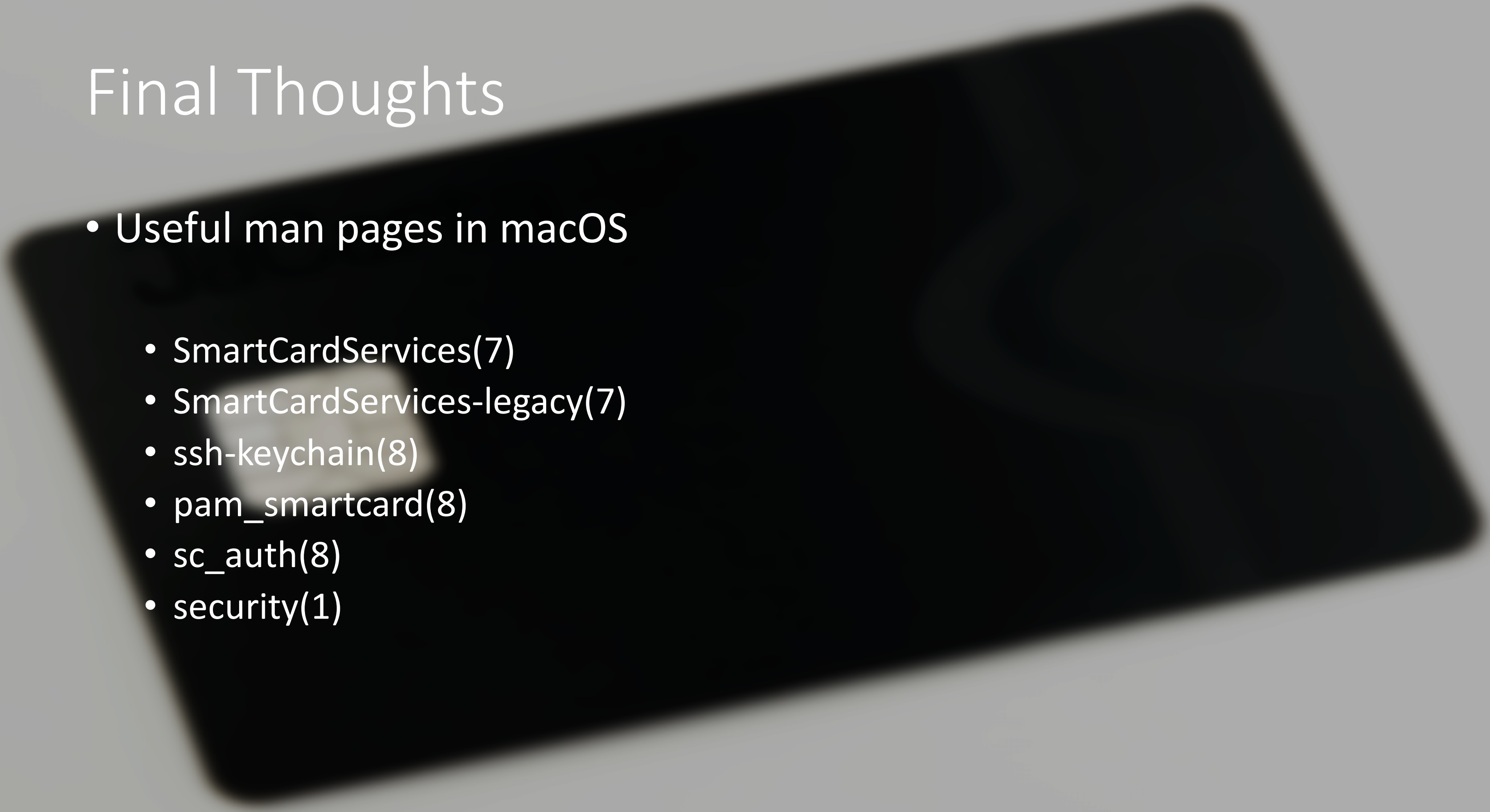
  - Can’t adequately defend against hardware hacks



Final Thoughts

- The general Mac trend is away from Directory Services
  - PKINIT cert authentication = No more keychain password issues
  - WOOHOO!
- I’m leaving a LOT out here.
  - Apple has been changing/implementing features with every point release
  - IdP’s such as Okta and Ping are implementing PIV support for SSO.
  - As long as the cert appears as part of the keychain, it’s available to all apps.

  - Here’s the torch. Now its up to you!



Final Thoughts

- Useful man pages in macOS
  - SmartCardServices(7)
  - SmartCardServices-legacy(7)
  - ssh-keychain(8)
  - pam_smartcard(8)
  - sc_auth(8)
  - security(1)

Big Thanks (and big links)

Big Thanks!

Todd Thoule
Yoann Gini
Callum Dean
Ludovic Rousseau

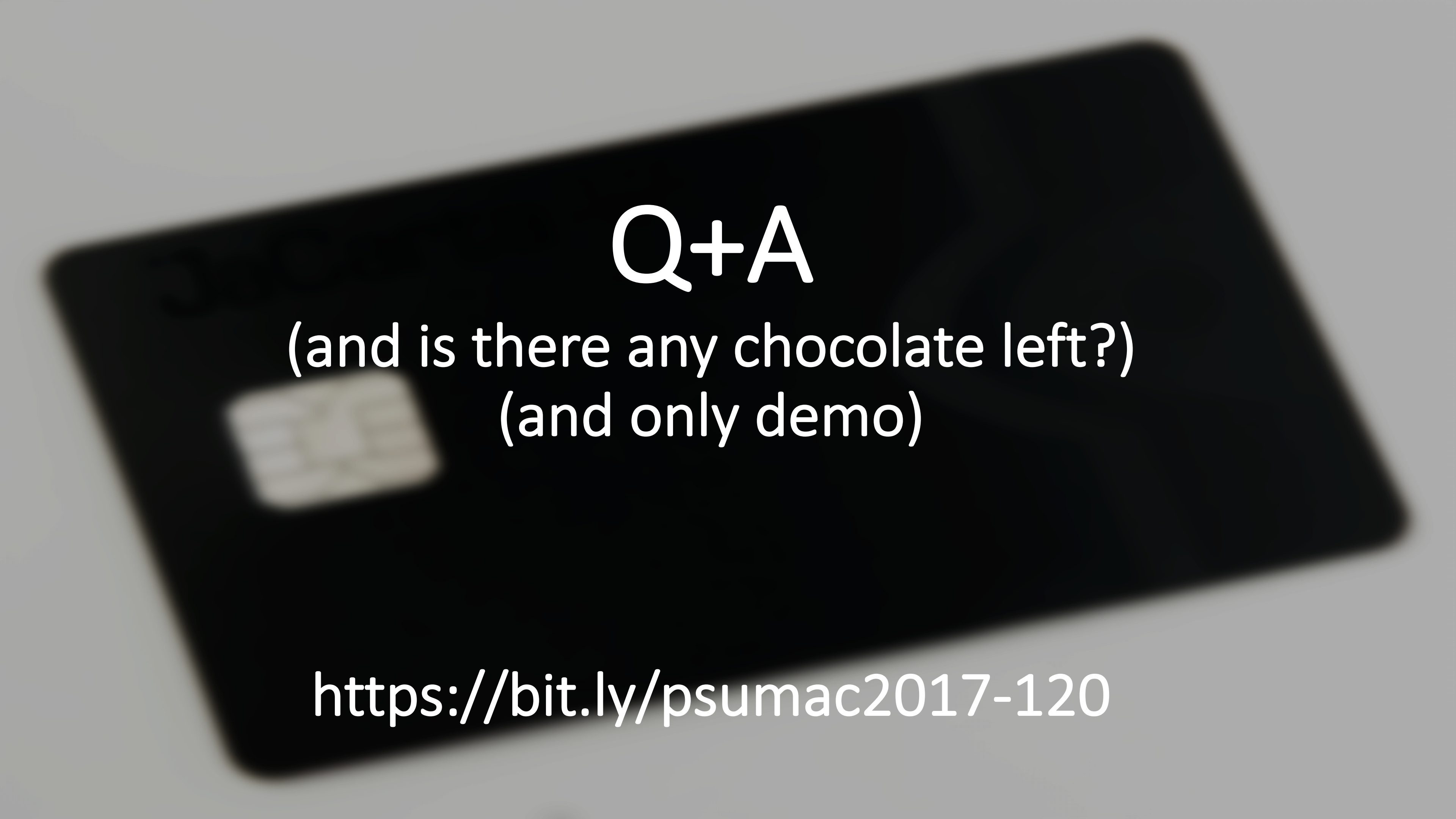
Daniel Hoit
Robert Hammen
Tom Bridge
Tom Witkowski
Shawn Geddis

Pepijn Bruienne
Dan Brodjieski
Owen Pragel
Joel Rennich

Everyone in #smartcard on Mac Admin Slack!

Big Links!

- MacAdmin Podcast
  - http://podcast.macadmins.org/2016/10/11/episode-12-two-factors-enter/
- Ludovic Rousseau
  - http://ludovicrousseau.blogspot.com
- Random Oracle
  - https://randomoracle.wordpress.com/2015/01/16/smart-card-logon-for-os-x-part-i/
- Me!
  - http://www.richard-purves.com/



Q+A

(and is there any chocolate left?)

(and only demo)

https://bit.ly/psumac2017-120