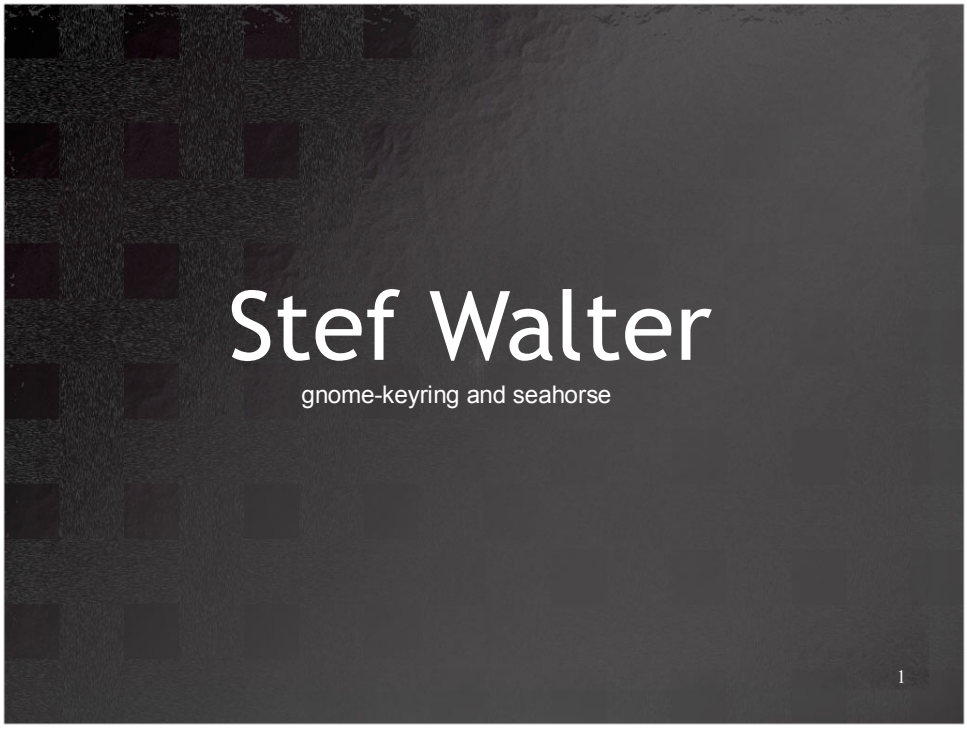


Stef Walter, maintainer of gnome-keyring and seahorse developer.

While we're getting setttled, a little about my day job.

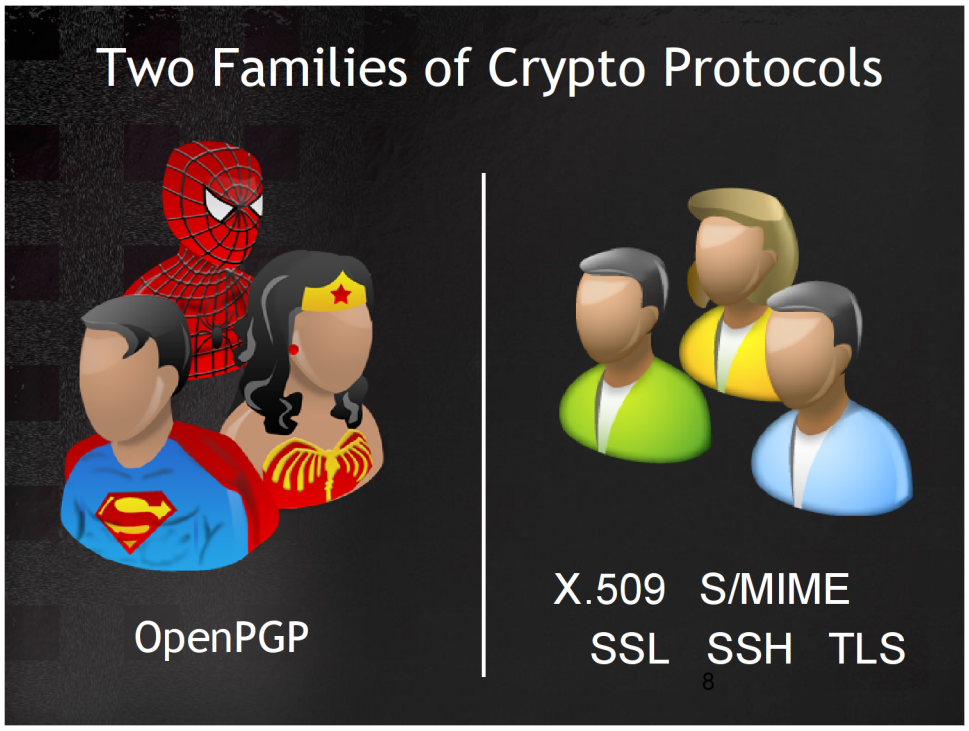
I work as a technical consultant and systems administrator for non-profits and missionaries.

Most notably, The Family International.

They do a great work all over the world, and I'm proud to help support it.

- Topic of today's talk is how we can lay a foundation for usable crypto in GNOME.
- Because everything is becoming more networked, and applications are increasingly running online, having strong, usable, and simple crypto support in the GNOME desktop is a must, to meet these needs.

- I'm interested in this topic because it brings together to seemingly conflicting requirements: usability and security.
- Users, companies and sometimes even governments expect developers to be miracle workers.
- There's the expectation of vault like security and doorknob like ease of use.
- It's a challenge to meet these expectations.
- We need to lay down a foundation of secure usable crypto in order to meet these expectations and challenges in GNOME applications.
- Credit where credit is due: There's been a lot of hard work on solid crypto implementations and libraries themselves. Here we're talking about making it usable.



- As we experience them in GNOME, there are two major families of crypto protocols:
- OpenPGP
- and “everything else”SSL, SSH, X.509 Certificates, S/MIME.
- OpenPGP sometimes gives you that 'crypto super hero' feeling.
- Other family is more mundane but no less important.
- Not here to discuss which is better: In GNOME we want to support both of these.

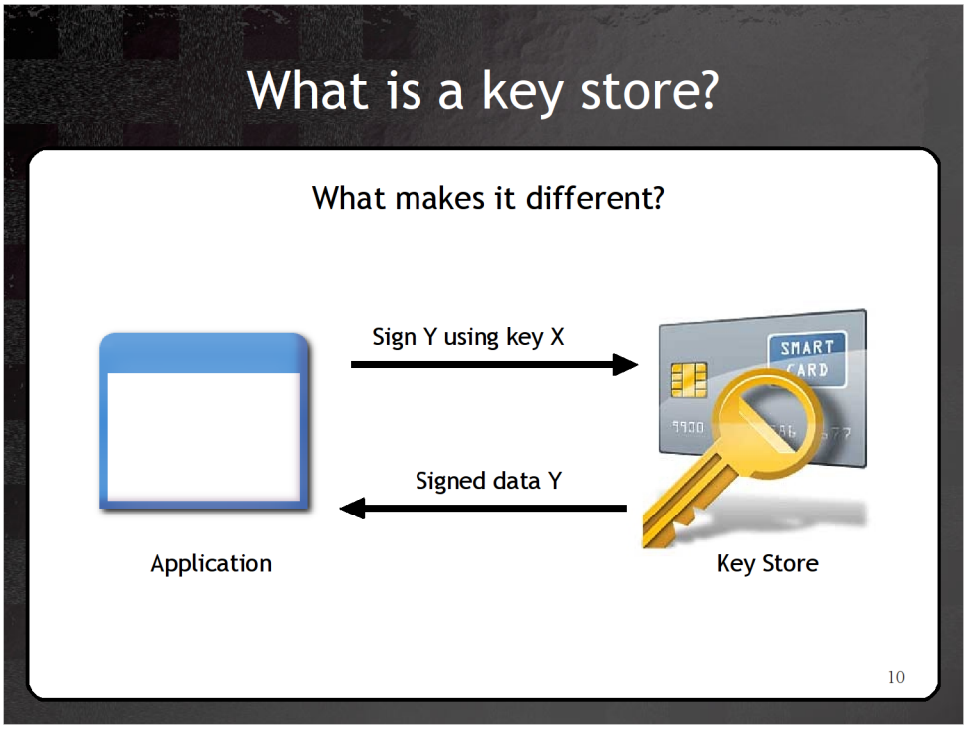
- What is a key store?

- There are lots of different places to store data: Databases, Filesystems, Settings (dconf)

- What makes it different than a database, or file system?

- The way that keys don't leave the store, and the store performs the operations. Example, smart card.

- Perform crypto operations using the keys, without the keys leaving the store.



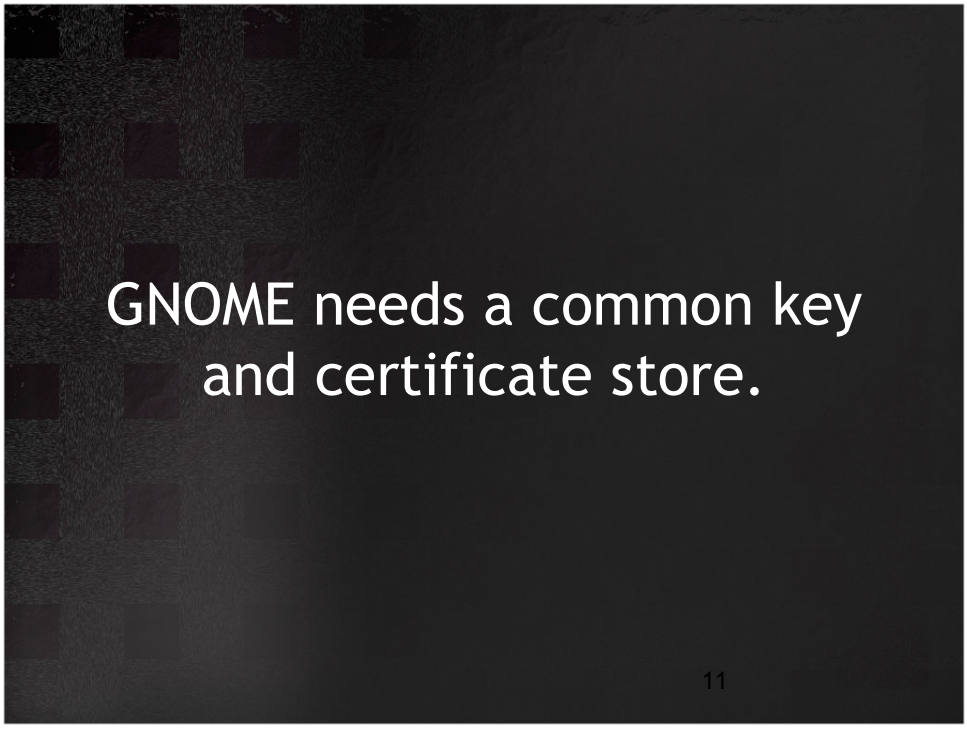
What is a key store?

There are lots of different places to store data: Databases, Filesystems, Settings (dconf)

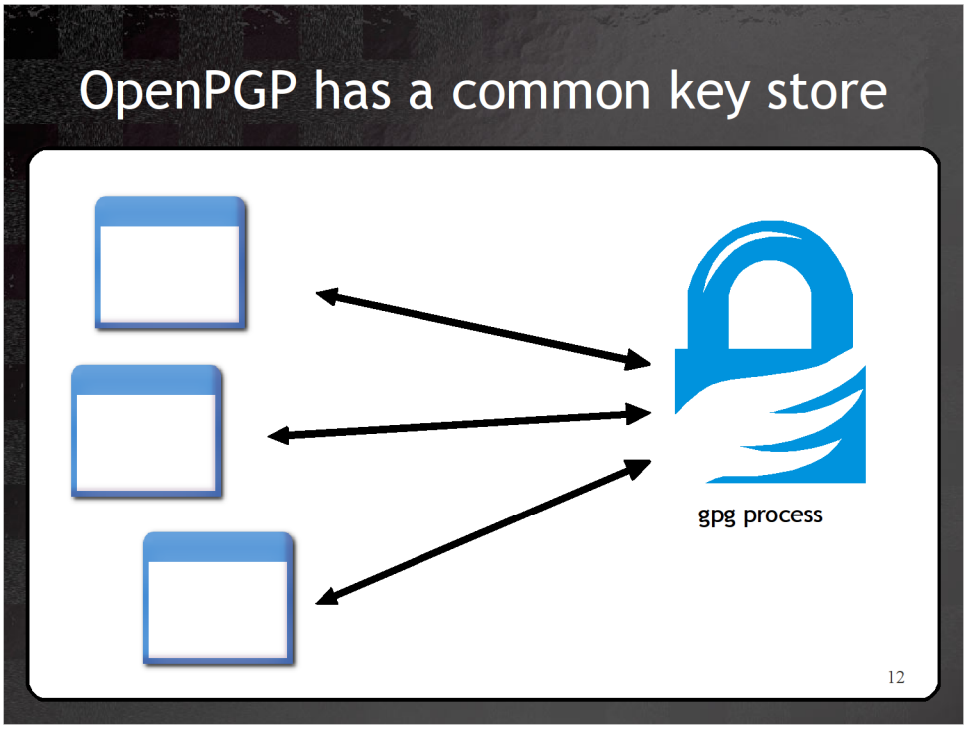
What makes it different than a database, or file system?

Perform crypto operations using the keys, without the keys leaving the store.

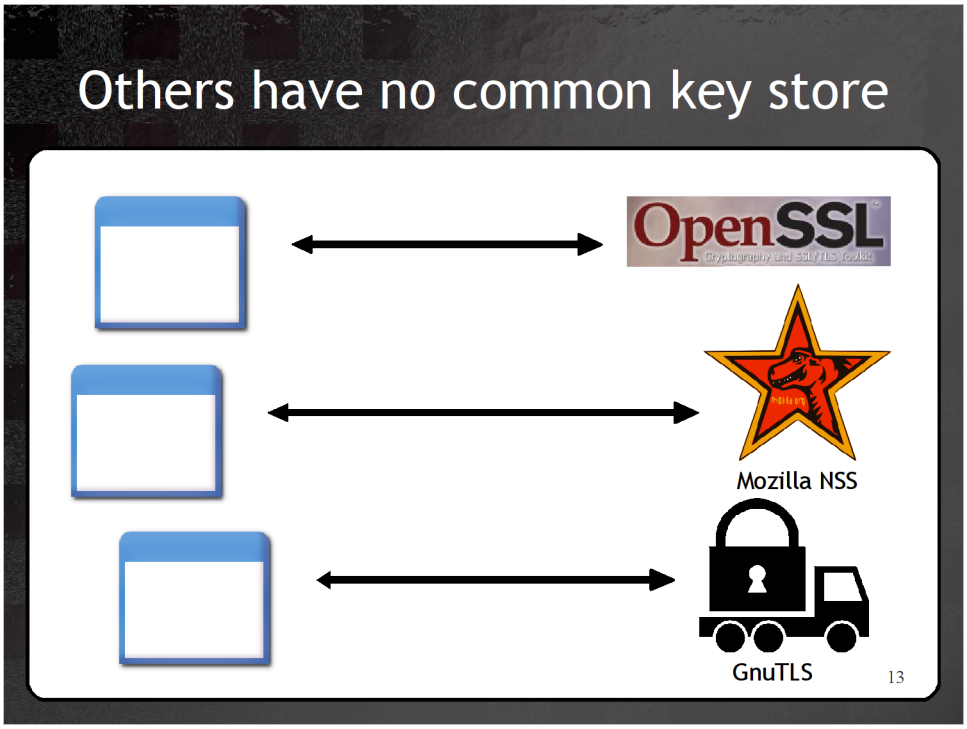
A: The way that keys don't leave the store, and the store performs the operations. Example, smart card.



- GNOME needs a common key and certificate store and user interface in order to be relevant with modern applications.
- Also common place to store CA root certificates for crypto that uses those.
- Lack of it hurts users, and application developers.

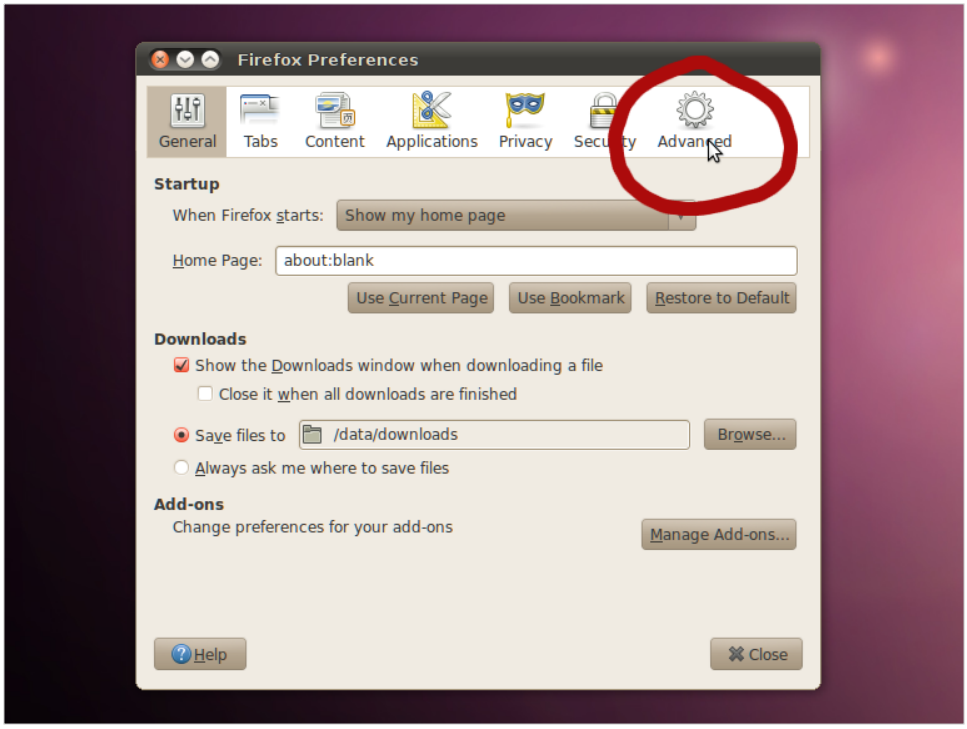


- On linux OpenPGP side already has a common 'key store'
- Called gnupg. You request that the gpg process encrypts or signs your data.
- The common 'key store' is somewhat solved for OpenPGP. Has some drawbacks, but solves problem of applications accessing and using keys.



- All the others (SSL, SSH, X.509 Certs etc..) don't have a common key store.
- Each application has to figure it out.
- In fact each of these libraries don't have a consistent place to store keys.

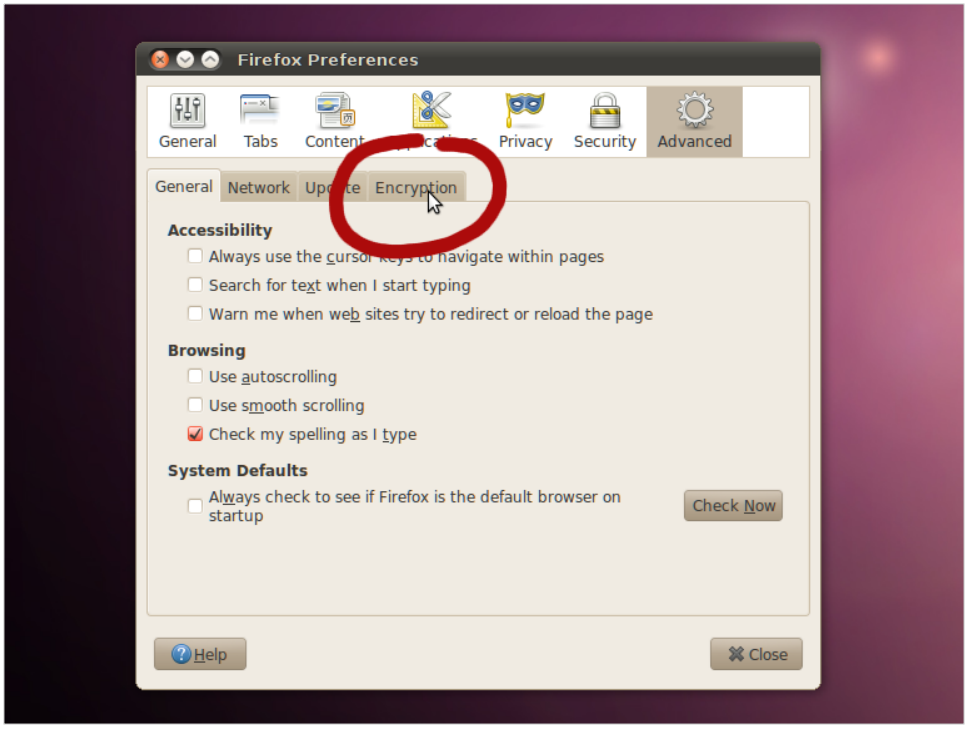
- Problems for the user:
- Using certificates in applications on the linux desktop is arcane and confusing.
- Users have to transfer keys manually from one application to another.
- Have to setup Root CA certificates in each application.
- Here's an example of the pain. Requesting certificate using firefox, and using in network manager.



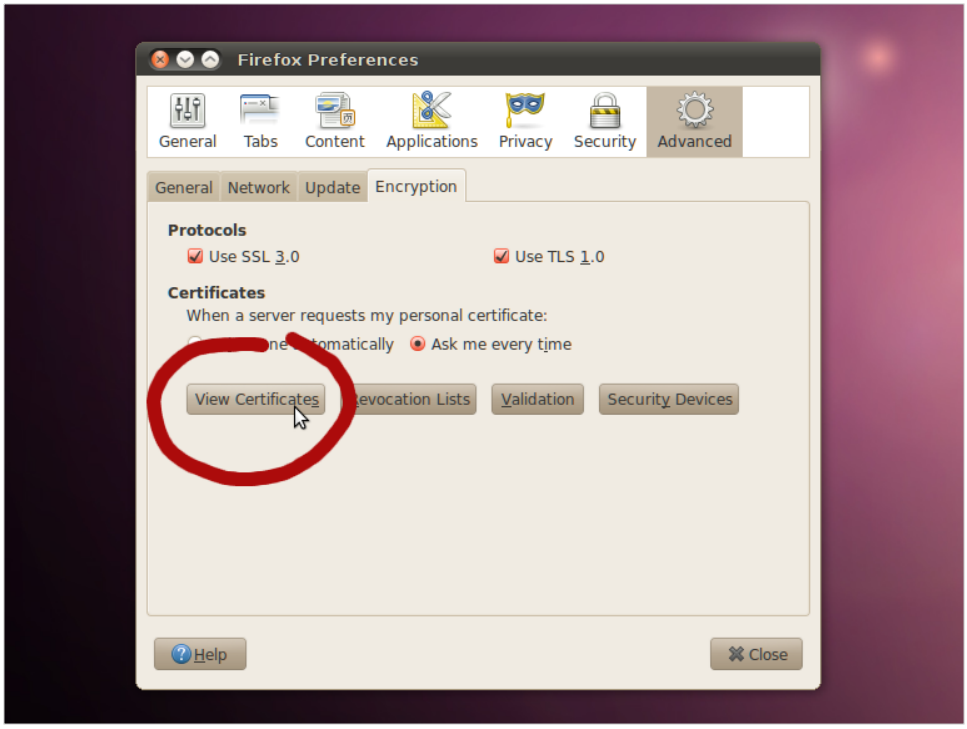
Example: User problems using Certificate with Network Manager

Request the certificate with firefox, firefox generates the key. We receive the certificate from the website.

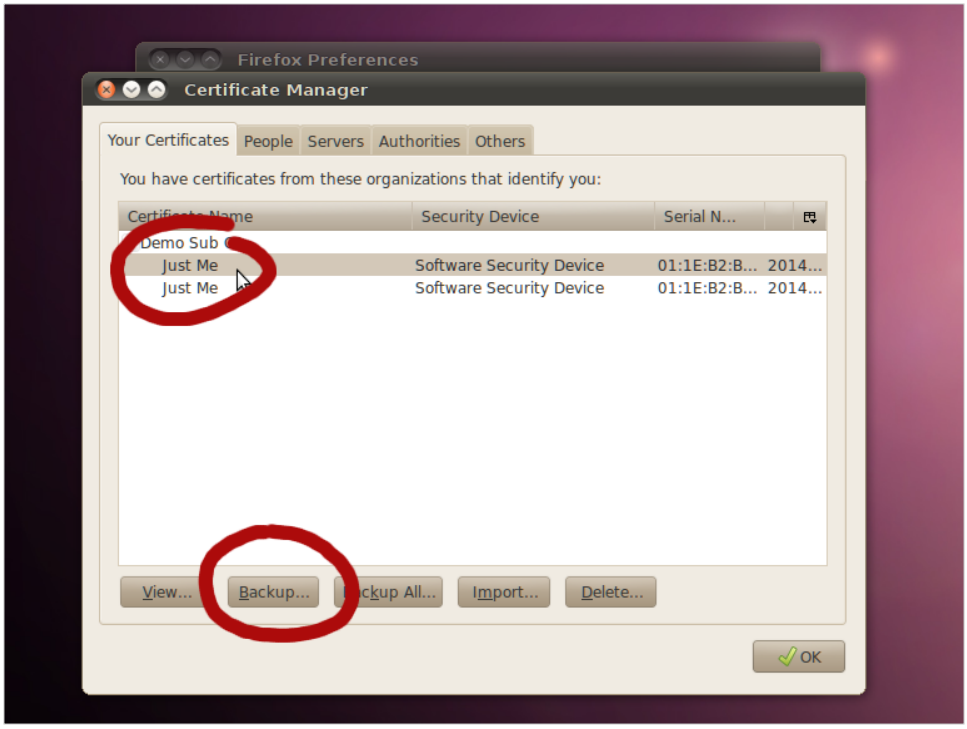
Firefox uses the NSS library, which stores it's keys in a private location only accessible by a single process.

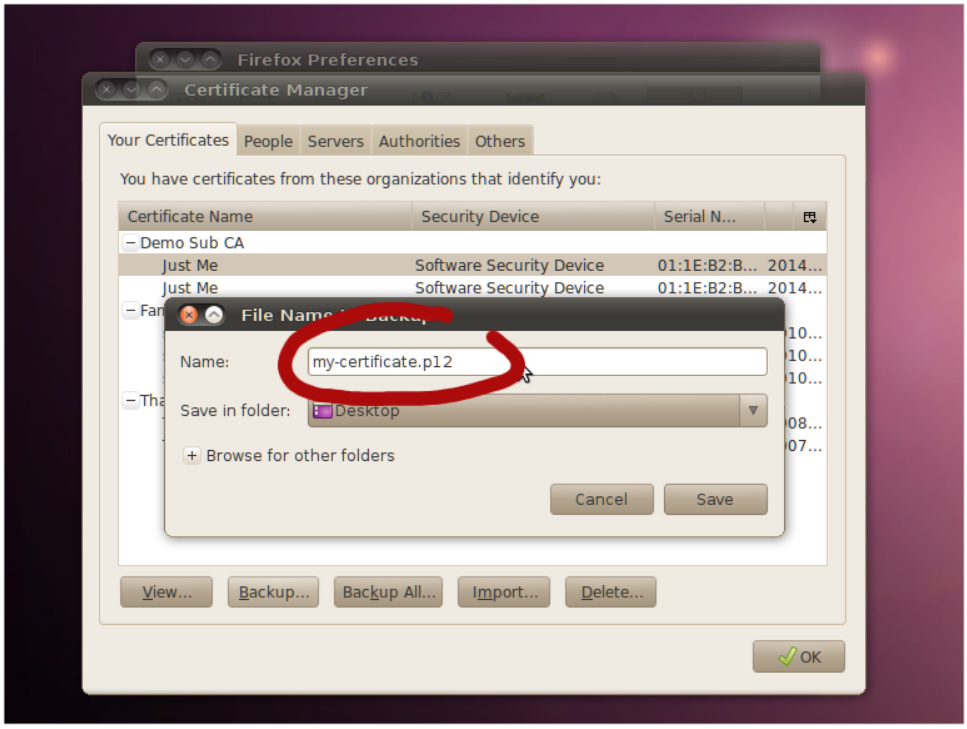


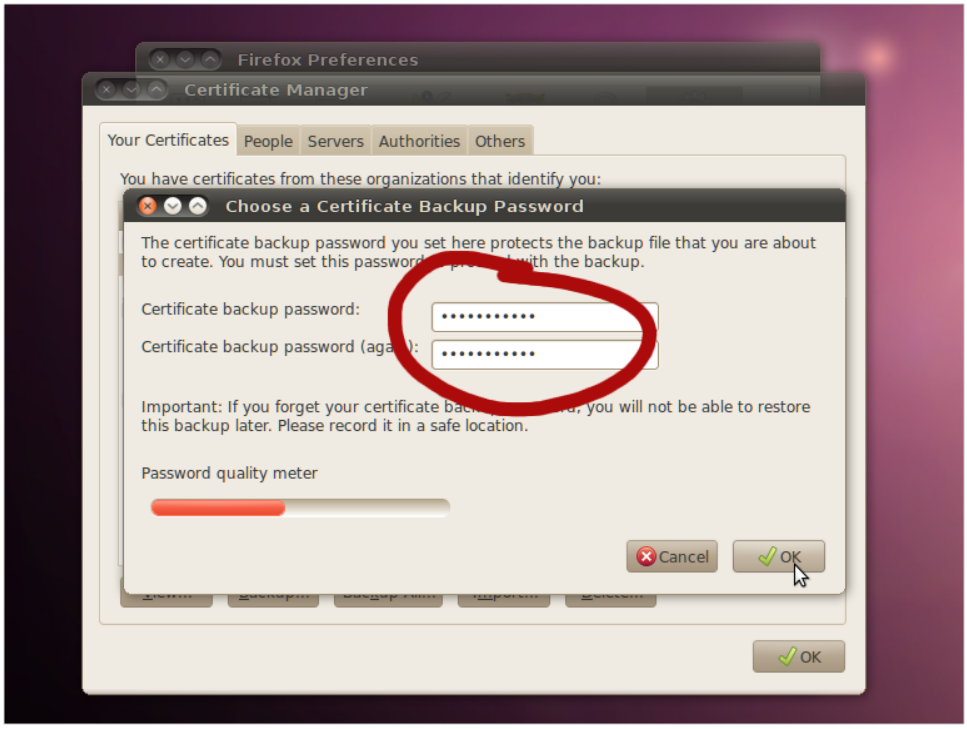
Dig through the bowels of firefox preferences, to find it your new certificate.

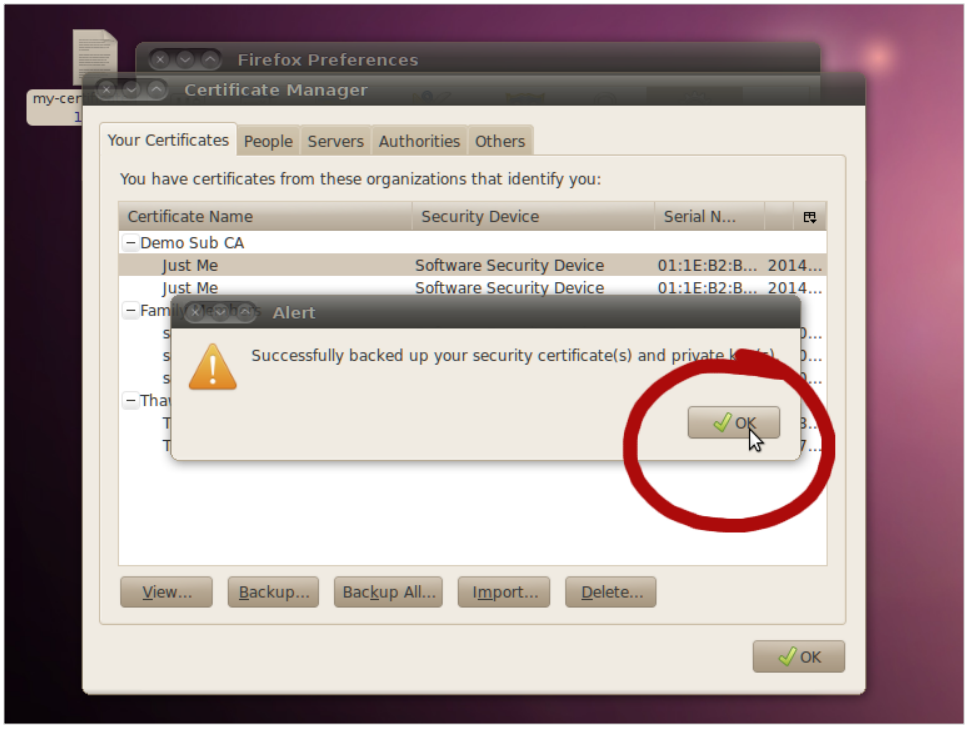


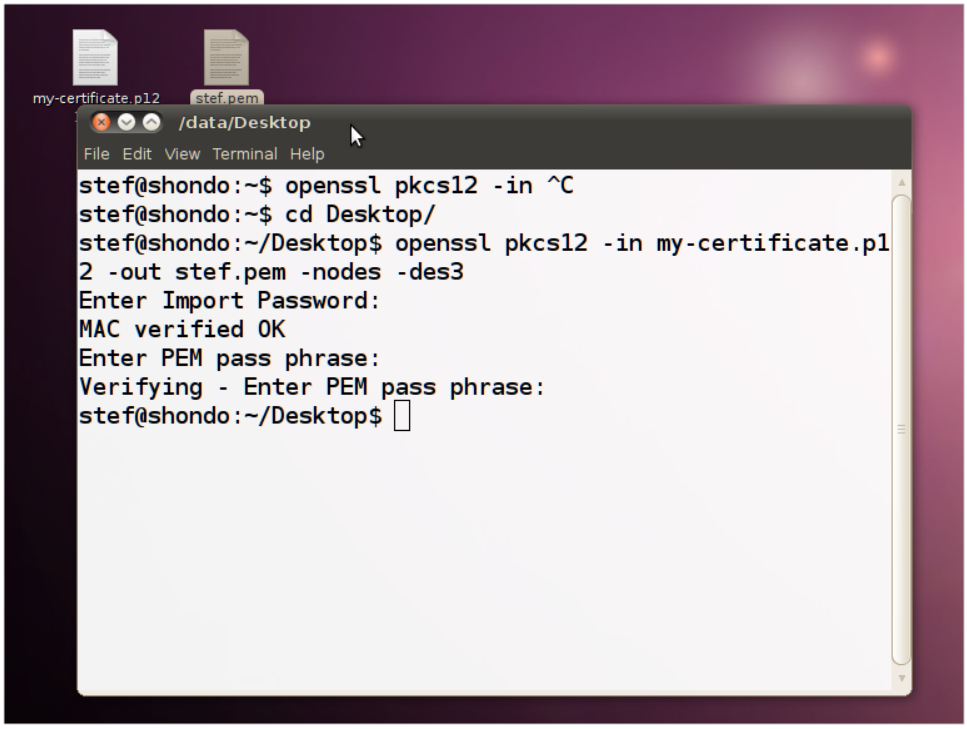
Export the key and certificate together in a single file. Export the CA????


Firefox Preferences
Certificate Manager
Your Certificates
People
Servers
Authorities
Others
You have certificates from these organizations that identify you:
Certificate Name
Security Device
Serial N...
Demo Sub
Just Me
Software Security Device
01:1E:B2:B...
2014...
Just Me
Software Security Device
01:1E:B2:B...
2014...
View...
Backup...
Backup All...
Import...
Delete...
OK


Firefox Preferences
Certificate Manager
Your Certificates
People
Servers
Authorities
Others
You have certificates from these organizations that identify you:
Certificate Name
Security Device
Serial N...
Demo Sub CA
Just Me
Software Security Device
01:1E:B2:B...
2014...
Just Me
Software Security Device
01:1E:B2:B...
2014...
File Name
Name:
my-certificate.p12
Save in folder:
Desktop
Browse for other folders
Cancel
Save
View...
Backup...
Backup All...
Import...
Delete...
OK


Firefox Preferences
Certificate Manager
Your Certificates
People
Servers
Authorities
Others
You have certificates from these organizations that identify you:
Choose a Certificate Backup Password
The certificate backup password you set here protects the backup file that you are about to create. You must set this password
with the backup.
Certificate backup password:
Certificate backup password (aga
):
Important: If you forget your certificate bac
d, you will not be able to restore this backup later. Please record it in a safe location.
Password quality meter
Cancel
OK
OK

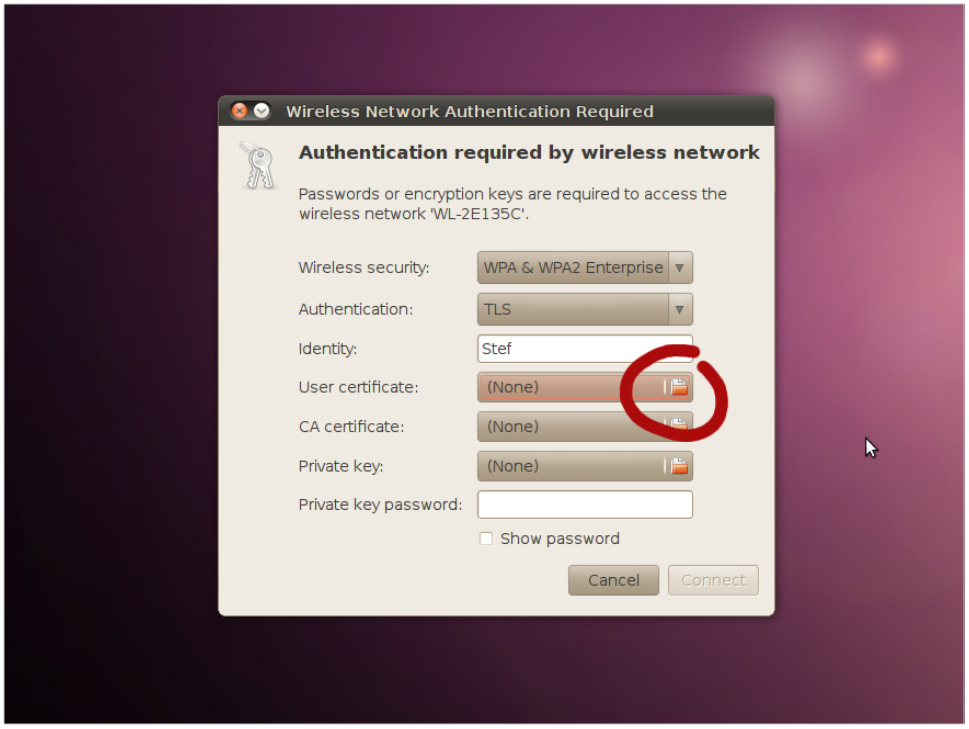

Firefox Preferences
Certificate Manager
Your Certificates
People
Servers
Authorities
Others
You have certificates from these organizations that identify you:
Certificate Name
Security Device
Serial N...
Demo Sub CA
Just Me
Software Security Device
01:1E:B2:B...
2014...
Just Me
Software Security Device
01:1E:B2:B...
2014...
Alert
Successfully backed up your security certificate(s) and private
OK
View...
Backup...
Backup All...
Import...
Delete...
OK

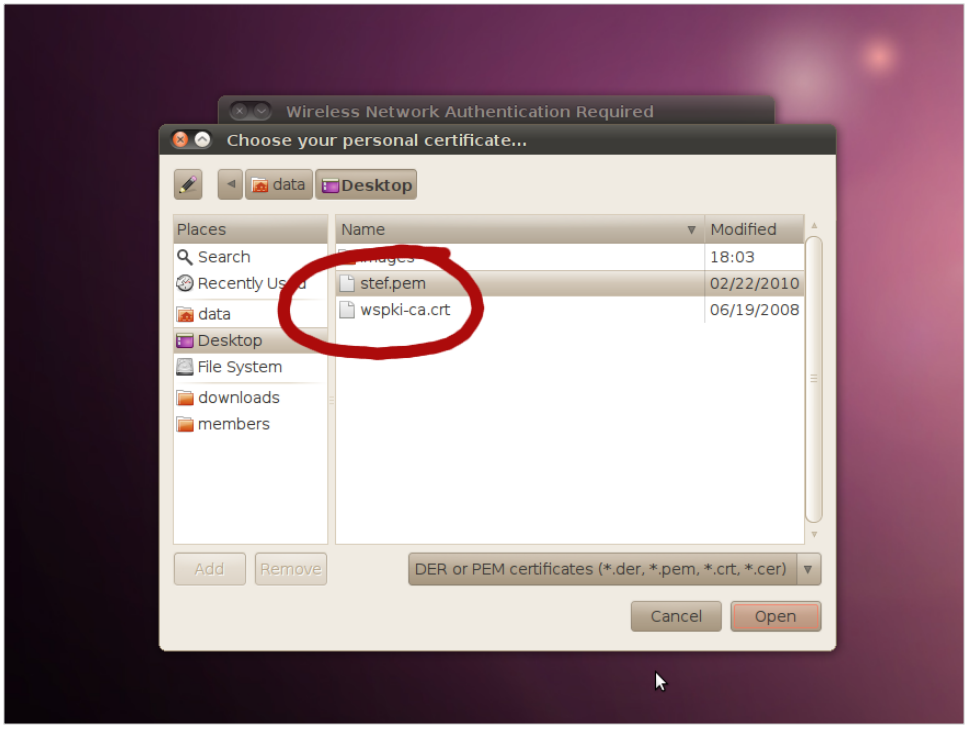


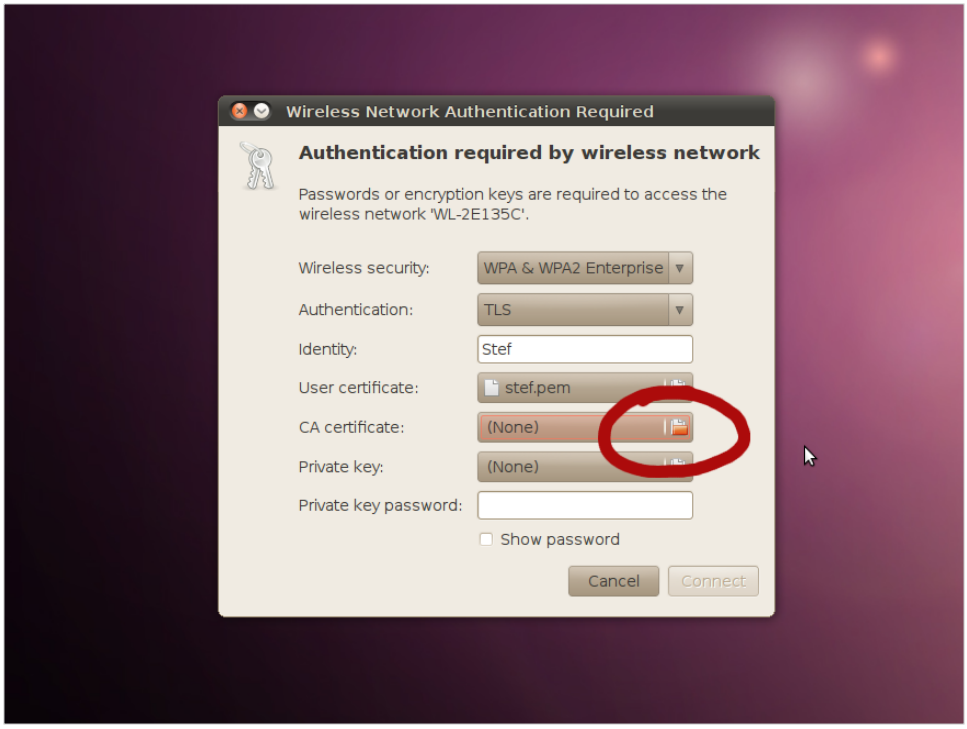
Network Manager uses OpenSSL expects a different format, with different files for key and certificate.

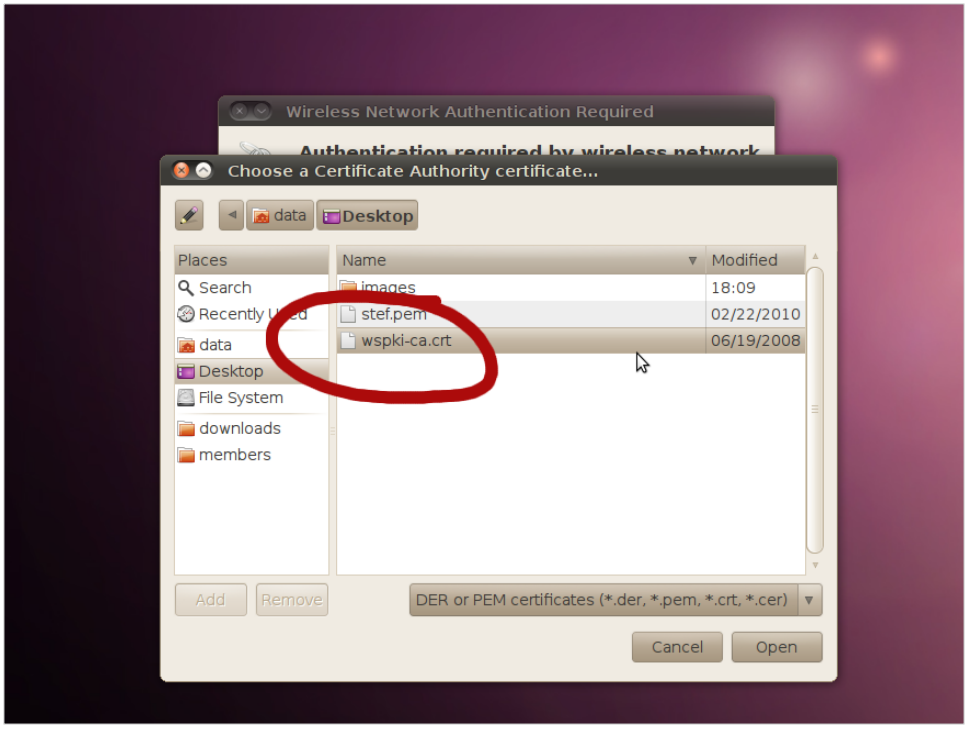
This is the command necessary to convert. Adds yet another password on the key.

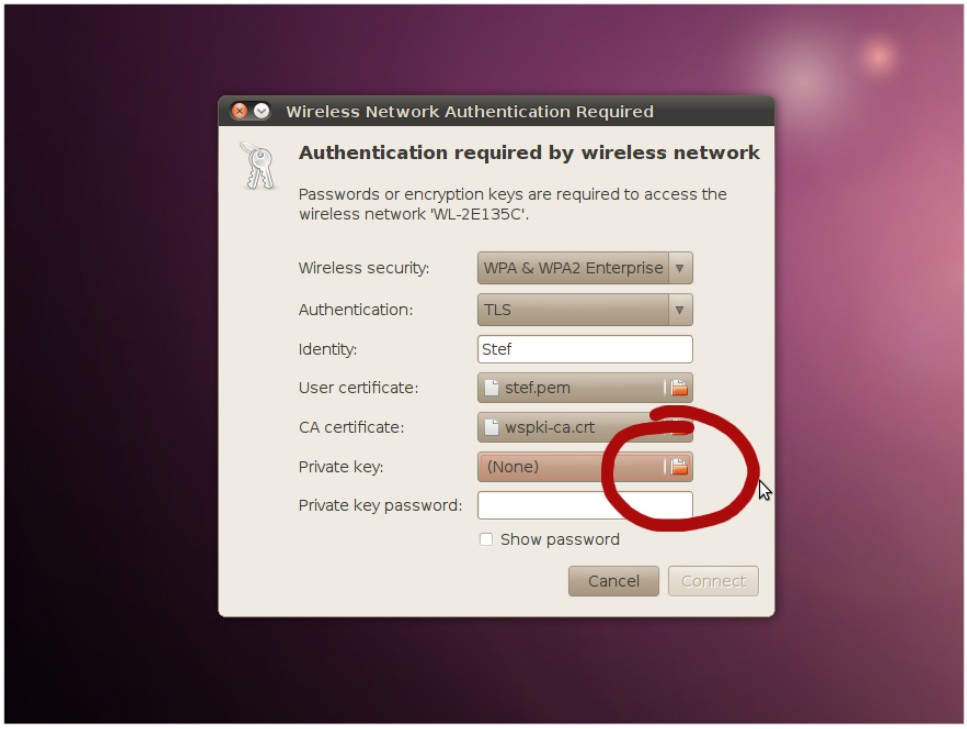
Point network manager at those files. Type the password that we just typed.

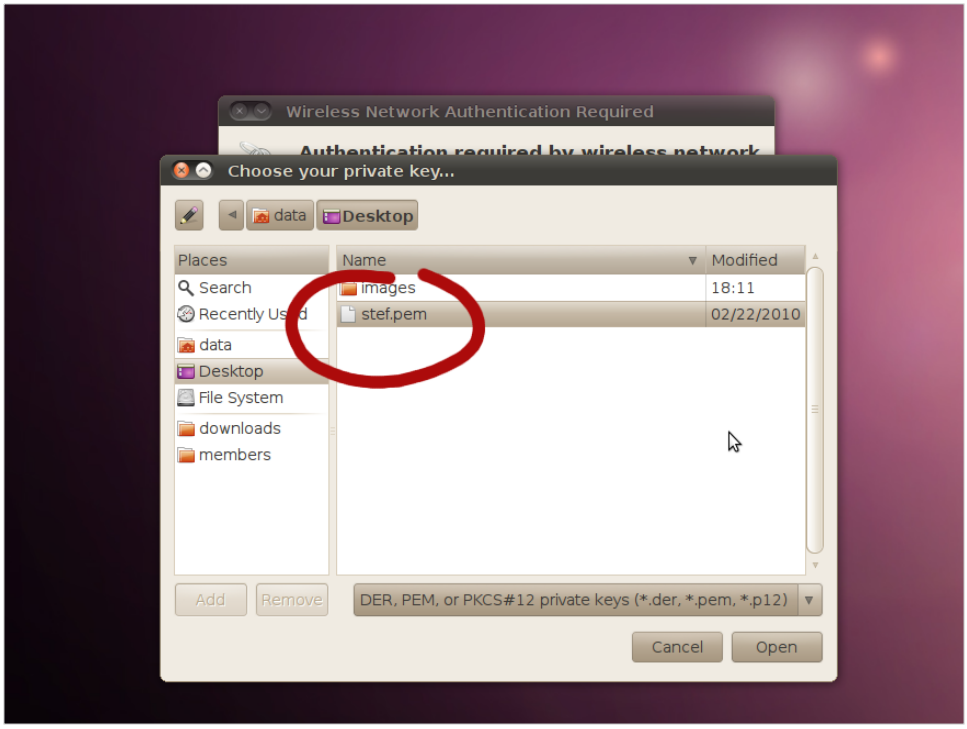

Wireless Network Authentication Required
Authentication required by wireless network
Passwords or encryption keys are required to access the wireless network 'WL-2E135C'.
Wireless security:
WPA & WPA2 Enterprise
Authentication:
TLS
Identity:
Stef
User certificate:
(None)
CA certificate:
(None)
Private key:
(None)
Private key password:
Show password
Cancel
Connect

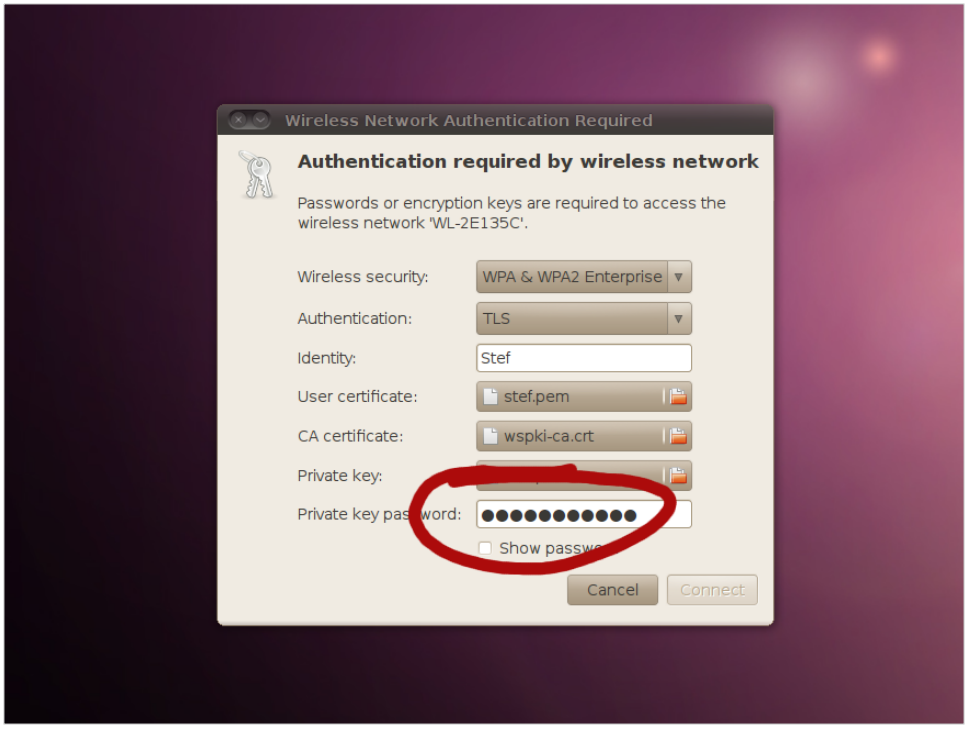

Wireless Network Authentication Required
Choose your personal certificate...
data
Desktop
Places
Search
Recently Used
data
Desktop
File System
downloads
members
Name
Modified
18:03
stef.pem
02/22/2010
wspki-ca.crt
06/19/2008
Add
Remove
DER or PEM certificates (*.der, *.pem, *.crt, *.cer)
Cancel
Open

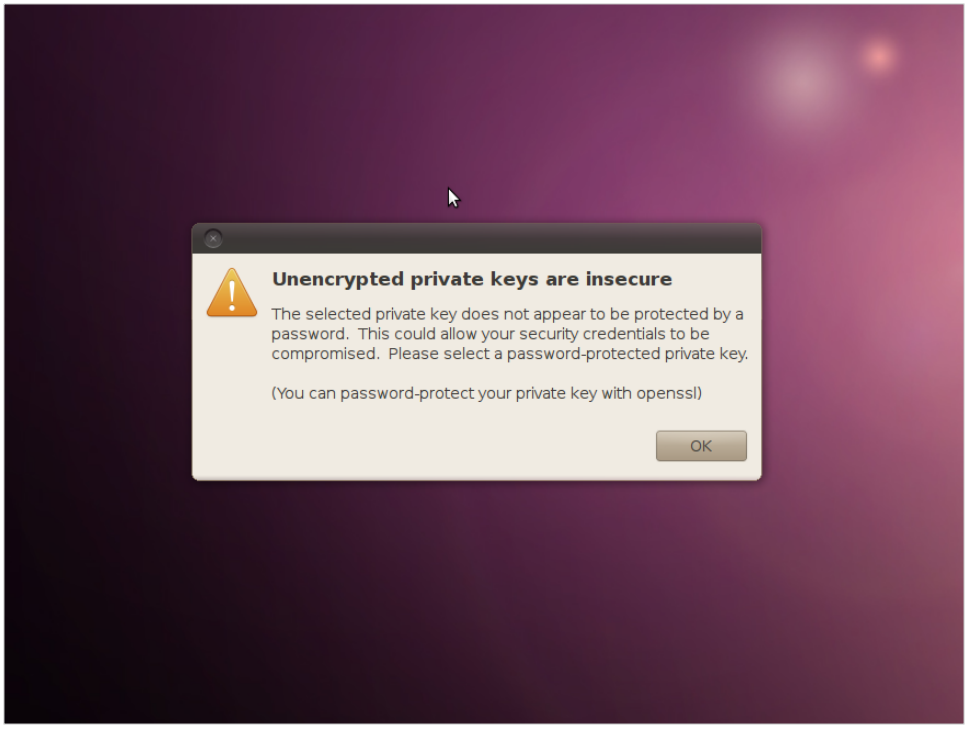

Wireless Network Authentication Required
Authentication required by wireless network
Passwords or encryption keys are required to access the wireless network 'WL-2E135C'.
Wireless security:
WPA & WPA2 Enterprise
Authentication:
TLS
Identity:
Stef
User certificate:
stef.pem
CA certificate:
(None)
Private key:
(None)
Private key password:
Show password
Cancel
Connect


Wireless Network Authentication Required
Choose a Certificate Authority certificate...
data
Desktop
Places
Search
data
Desktop
File System
downloads
members
Name
Modified
images
18:09
stef.pem
02/22/2010
wspki-ca.crt
06/19/2008
Add
Remove
DER or PEM certificates (*.der, *.pem, *.crt, *.cer)
Cancel
Open


Wireless Network Authentication Required
Authentication required by wireless network
Passwords or encryption keys are required to access the wireless network 'WL-2E135C'.
Wireless security:
WPA & WPA2 Enterprise
Authentication:
TLS
Identity:
Stef
User certificate:
stef.pem
CA certificate:
wspki-ca.crt
Private key:
(None)
Private key password:
Show password
Cancel
Connect

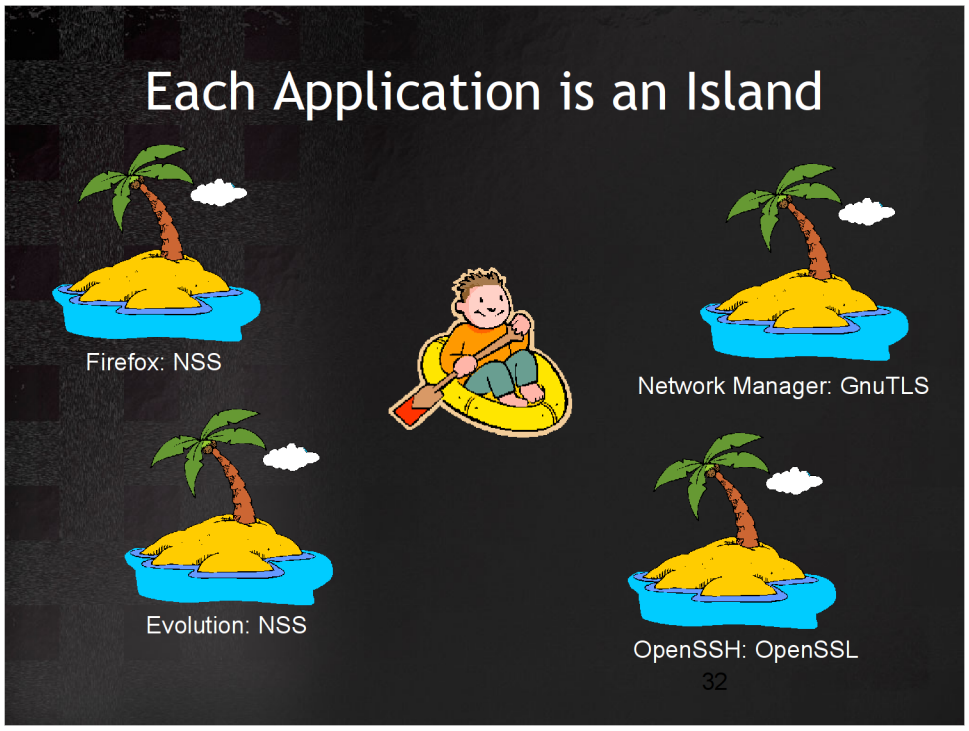

Wireless Network Authentication Required
Choose your private key...
data
Desktop
Places
Search
Recently Used
data
Desktop
File System
downloads
members
Name
Modified
images
18:11
stef.pem
02/22/2010
Add
Remove
DER, PEM, or PKCS#12 private keys (*.der, *.pem, *.p12)
Cancel
Open


Wireless Network Authentication Required
Authentication required by wireless network
Passwords or encryption keys are required to access the wireless network 'WL-2E135C'.
Wireless security:
WPA & WPA2 Enterprise
Authentication:
TLS
Identity:
Stef
User certificate:
stef.pem
CA certificate:
wspki-ca.crt
Private key:
Private key password:
Show passwo
Cancel
Connect

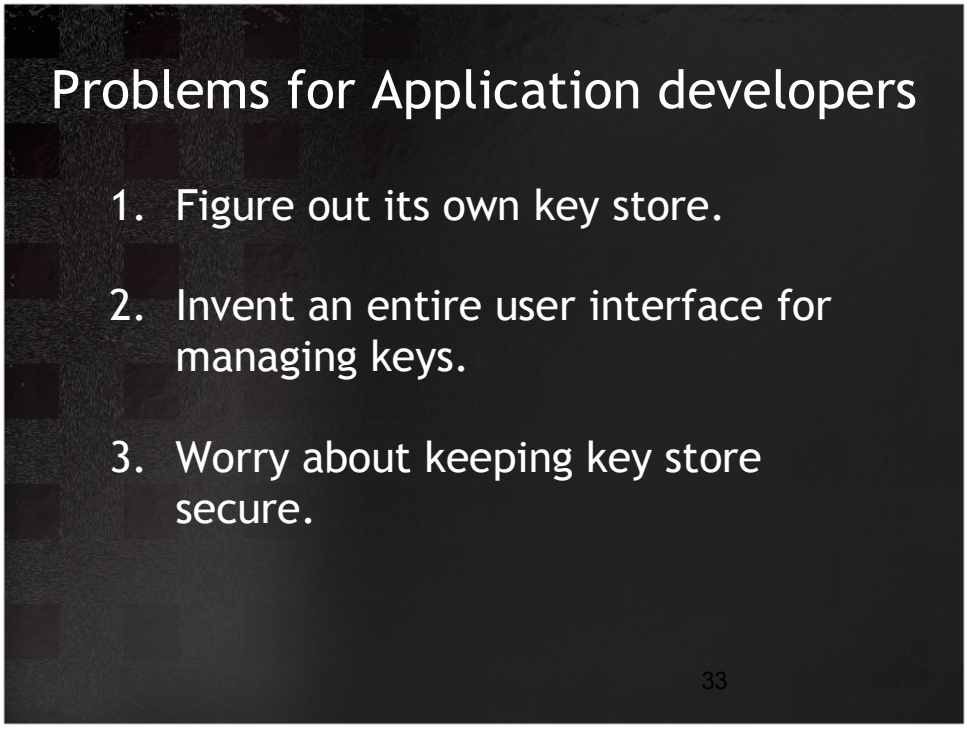

Unencrypted private keys are insecure
The selected private key does not appear to be protected by a password. This could allow your security credentials to be compromised. Please select a password-protected private key.
(You can password-protect your private key with openssl)
OK

Many people might say that nobody uses this stuff, and now we see why.

Same sort of fail applies to trying to import your Trusted Root CA into every application.



- Each application is an Island: Requires the user to ferry keys and certificates between them.
- We have many crypto libraries used in GNOME: OpenSSL, NSS, GnuTLS
- None of these libraries even have a standard place to store their keys and certificates.
- It's good that applications are not locked into a specific store for their keys and certificates. This allows flexibility in design and security requirements.
- However it's not good that the user is relegated to the role of janitor, cleaning up after programs.



- Problems for application developers.
- Each application has to bother decide about key storage and formats.
- Each application has to build its own interface.
- Each application has to worry about keeping keys secure.
- Each application has to find out where they can look up any CA certificates that they need.
- Firefox for example has to have a master password set, which you type in every time, otherwise stores keys in cleartext on disk.
- Cannot use smart cards or other devices if with applications.
- **New Applications?** Example: Encrypted backups. Example: Encrypted IM conversations.

- Can we fix it?
- Yes we can!

- Diversity is great. Having multiple desktops, OS's.

- Allows each application to meet own security requirements.

- Some Distros need 'FIPS' others need 'GNU/Freedom'

- Different licenses, styles, personalities.

- Encourages progress.

- It's great as long as we have standards.

- One library like NSS everywhere? We can do better and still allow NSS to meet their needs and requirements.

- We can have our cake and eat it too. By having a standard for key and certificate storage.

- PKCS#11 is a API that allows an application to use keys in a key storage like a smart card.

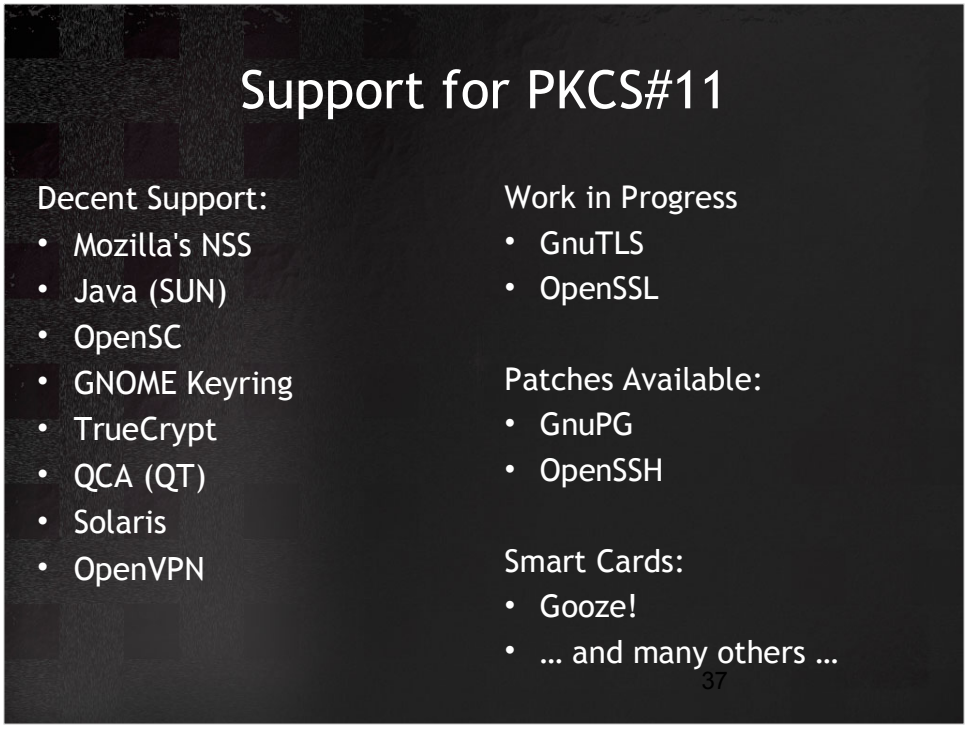
- In fact it was designed for smart cards.

- It's a bit old school.

- In theory PKCS#11 isn't the necessarily the best API possible. It has some warts and problems, missing bits.

- But the important thing it's a standard implemented in lots of places, that allows us to solve our “island” problem.

- We're using the key and object storage part of PKCS#11.



- Support for PKCS#11 in various libraries.

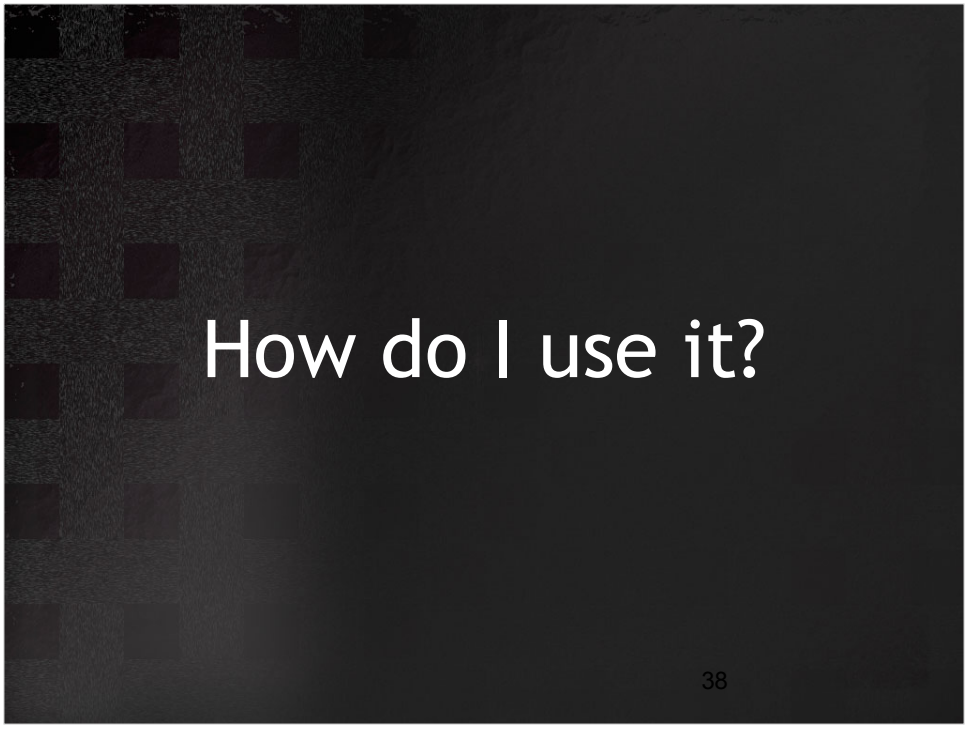
- NSS has great complete support for PKCS#11.

- GnuTLS PKCS#11 is being actively worked on. Been in touch with developers.

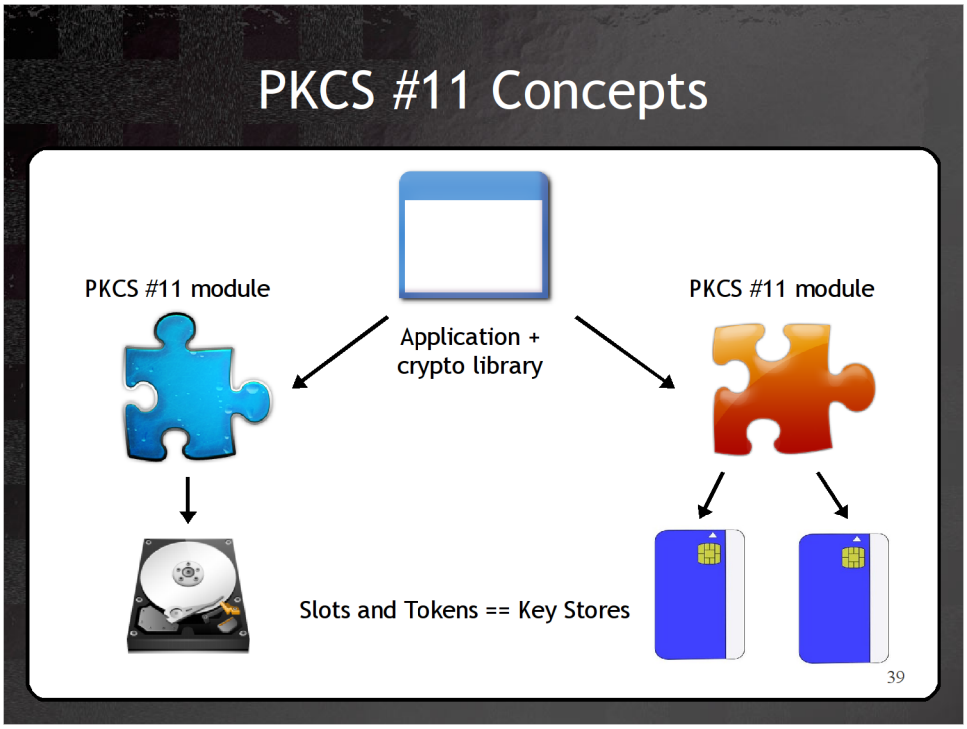
- OpenSSL has a pluggable engine for PKCS#11.

- Gnome Keyring is built around PKCS#11. Even the SSH agent is built around PKCS#11.

- Just spoke with Nikos, about GnuTLS implementation.



- If you're not developing an application that manages keys, with any luck you won't have to deal with PKCS#11 directly.
- But you need to understand the infrastructure that it enables.
- Depending on the crypto library you use, you may 'experience' more or less of PKCS#11.



Common ground between applications and libraries.

It's defined as a bare bones set of C function pointers.

Specifies application programming interface (“Cryptoki”) in C

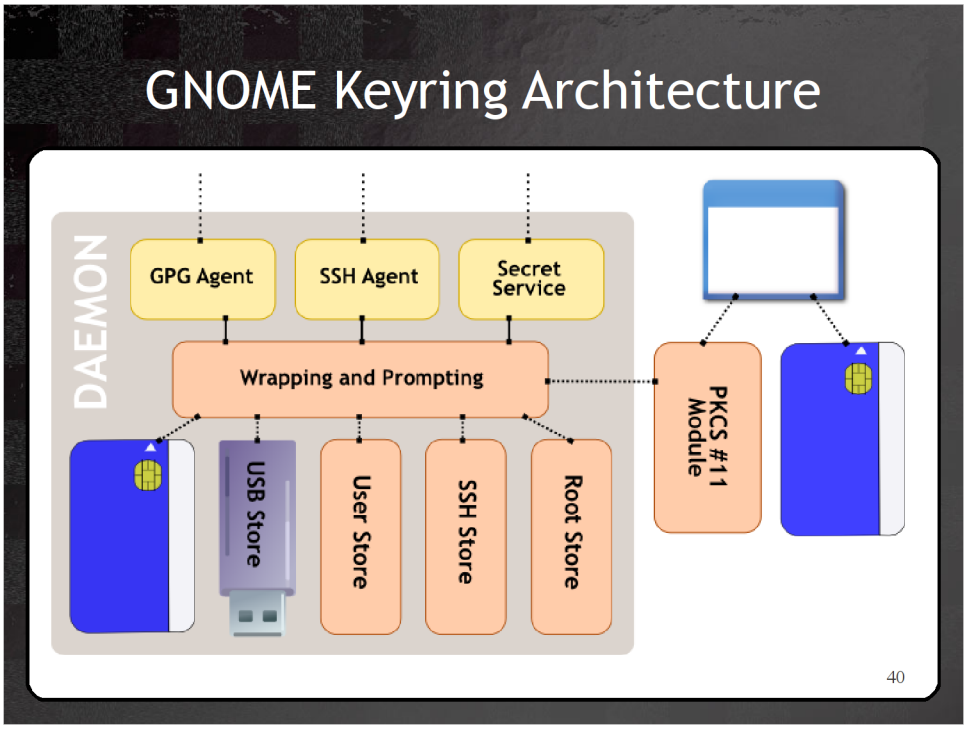
Objects (such as certificates or keys) may be created, read, updated and deleted

But more importantly it defines how to do crypto operations on keys without the keys leaving the storage.

An application loads one or more PKCS#11 modules, each of which allow access to a different kind of key storage.

Slots and tokens.

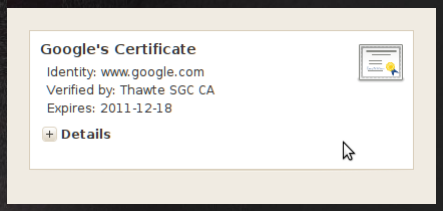
TODO: More details about slots, tokens and objects.



- Here's an overwiew of the gnome keyring architecture.
- External drivers (like smart cards) are loaded directly by the application and by the daemon, since they're not designed to be remoted.
- The GPG, SSH agents use PKCS#11.
- Applications load a PkCS#11 module which then communicates with daemon.

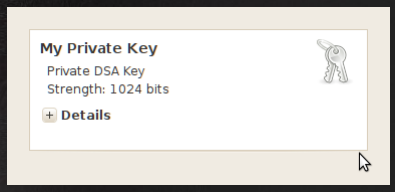
- Much user interface dissapears since the user is no longer janitor.
- Mention libcryptui and work on the OpenPGP side.

Certificate Display

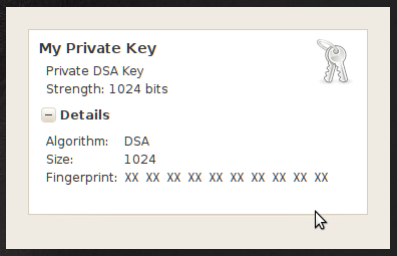


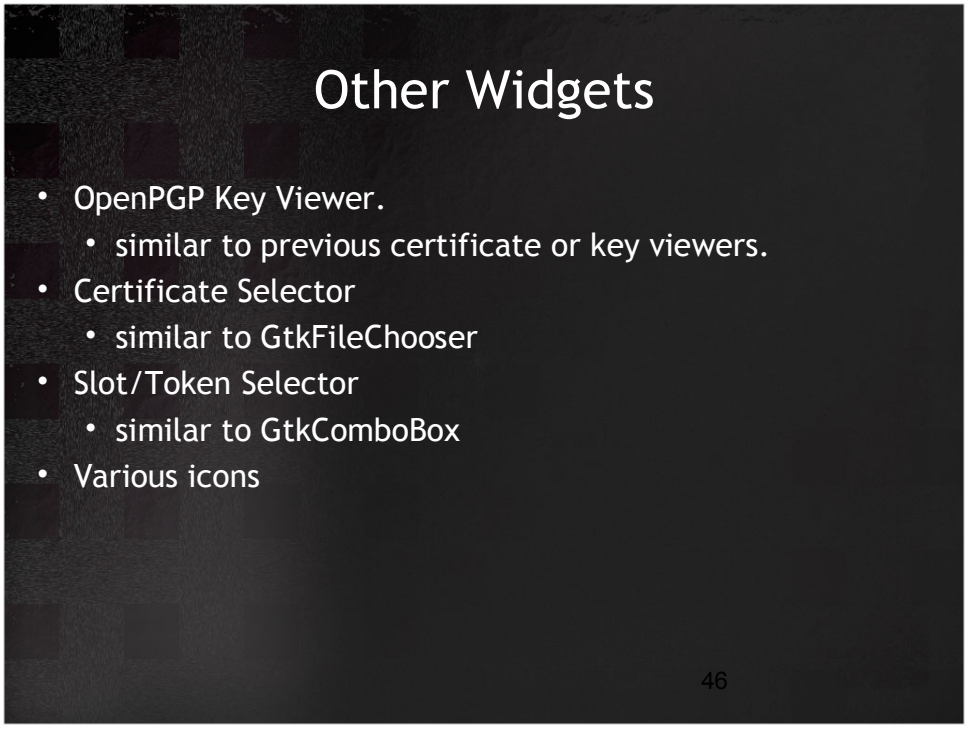
Certificate Display

Key Display



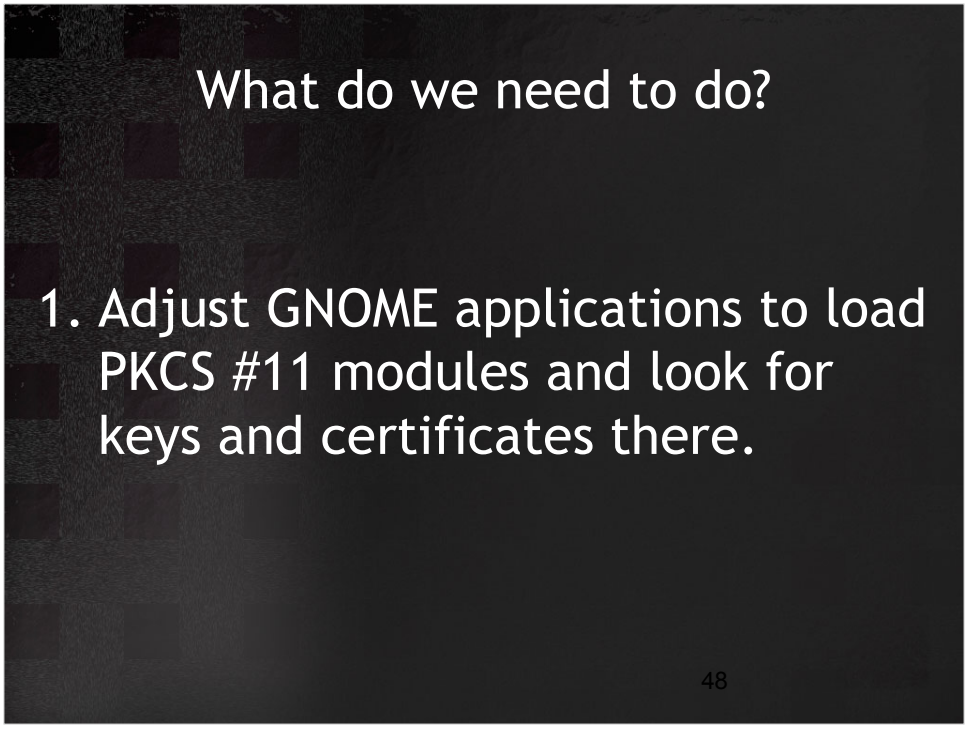
Key Display


My Private Key
Private DSA Key
Strength: 1024 bits
Details
Algorithm: DSA
Size: 1024
Fingerprint: XX XX XX XX XX XX XX XX XX XX

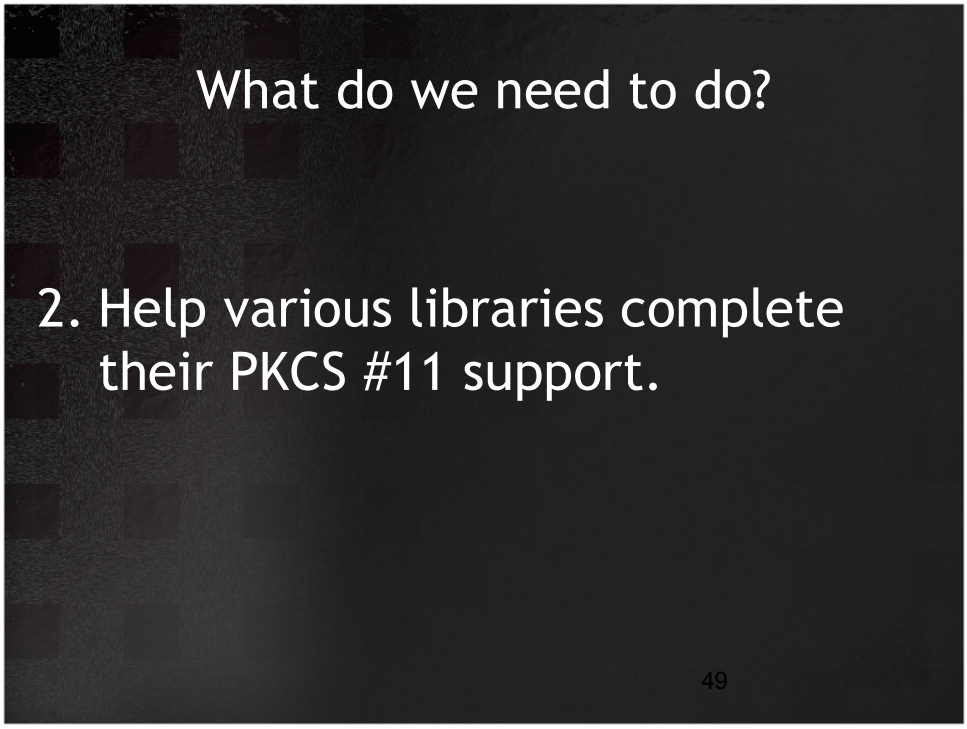


Certificate selector used in network manager.

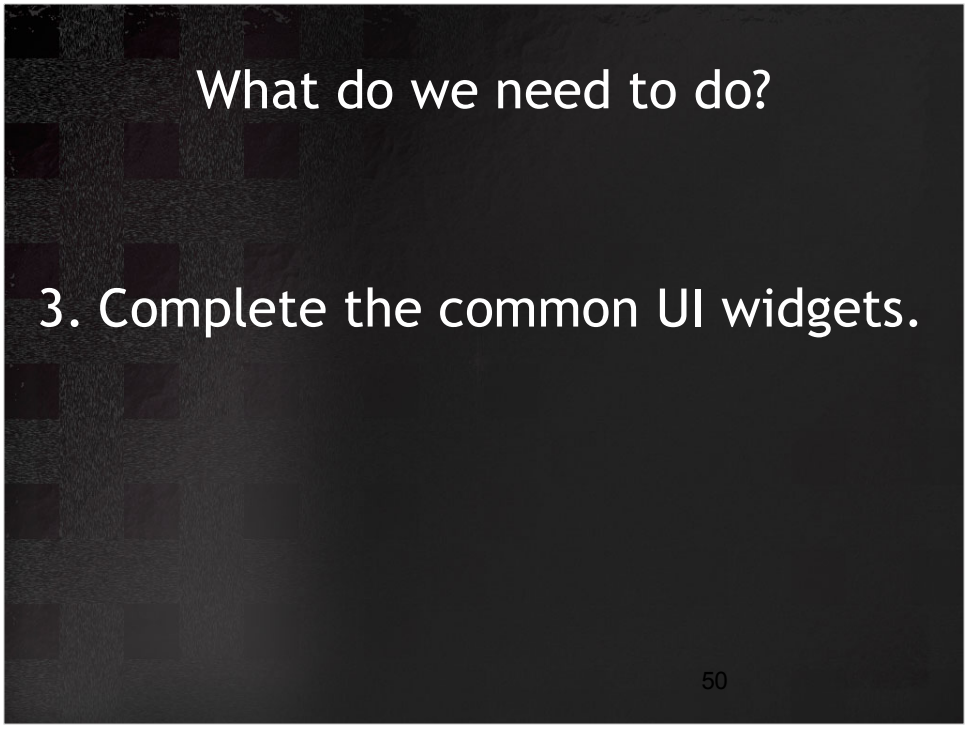
- What can you do right now?
- Use an SSH agent which gets its keys from gnome-keyring and PKCS#11
- Use your web browser (like firefox) with CA certificates dropped into /etc/ssl/certs
- Use your web browswer or email client with client certificates stored in gnome-keyring.



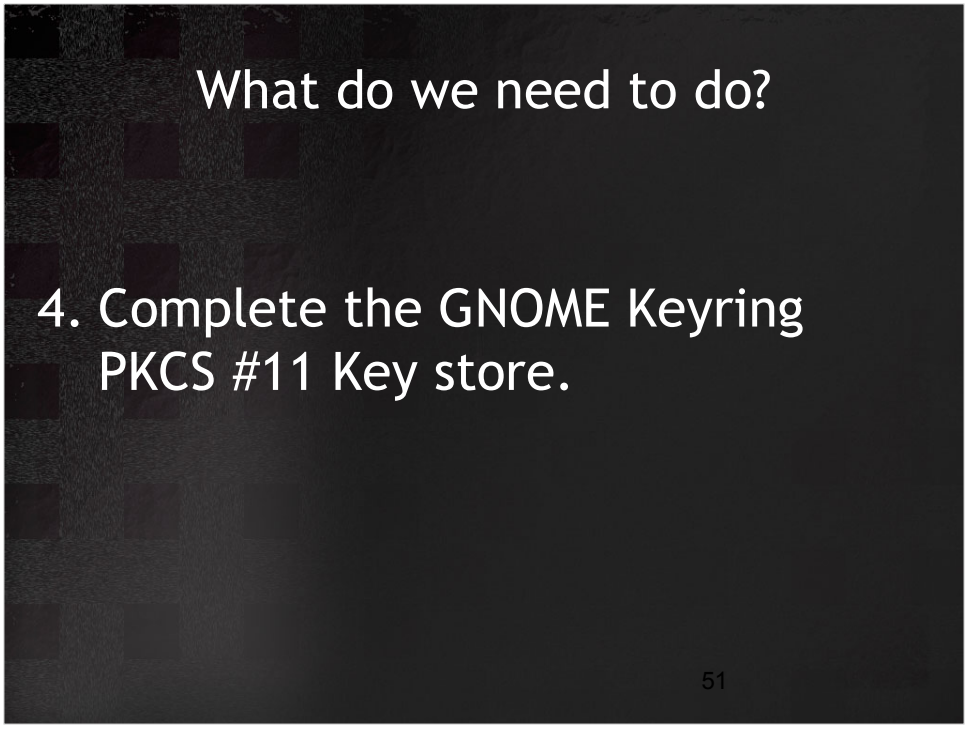
- Probably make a small helper library, perhaps together with the UI widgets.
- If you get patches for your project this is what it's about.



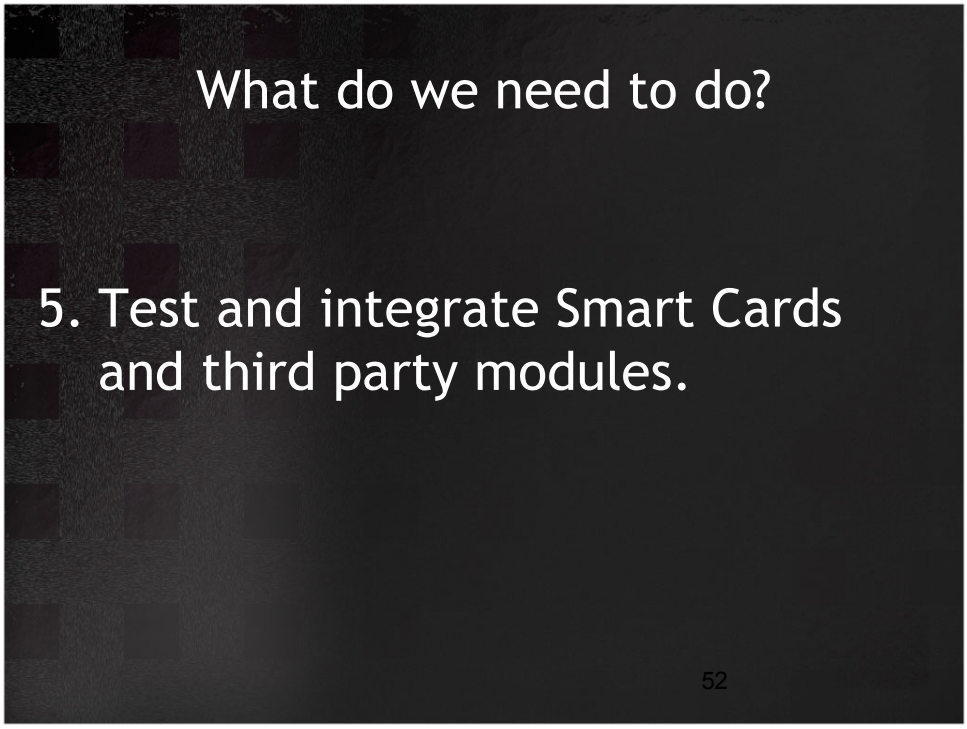
- GnuTLS guys working hard.
- Working on spec for specifying which PKCS #11 modules to load.
- Similar to PAM config files?
- Extension to PKCS#11 on marking trusted CA roots.



- There's already a library called libgcr (part of gnome-keyring).
- Needs more work to complete the UI widgets.
- Take best ideas of seahorse's libcryptui.



- Better smart card support.
- Lots of little corner cases need completion.
- Example key generation.
- Example .ssh write support.
- Support for removable storage.

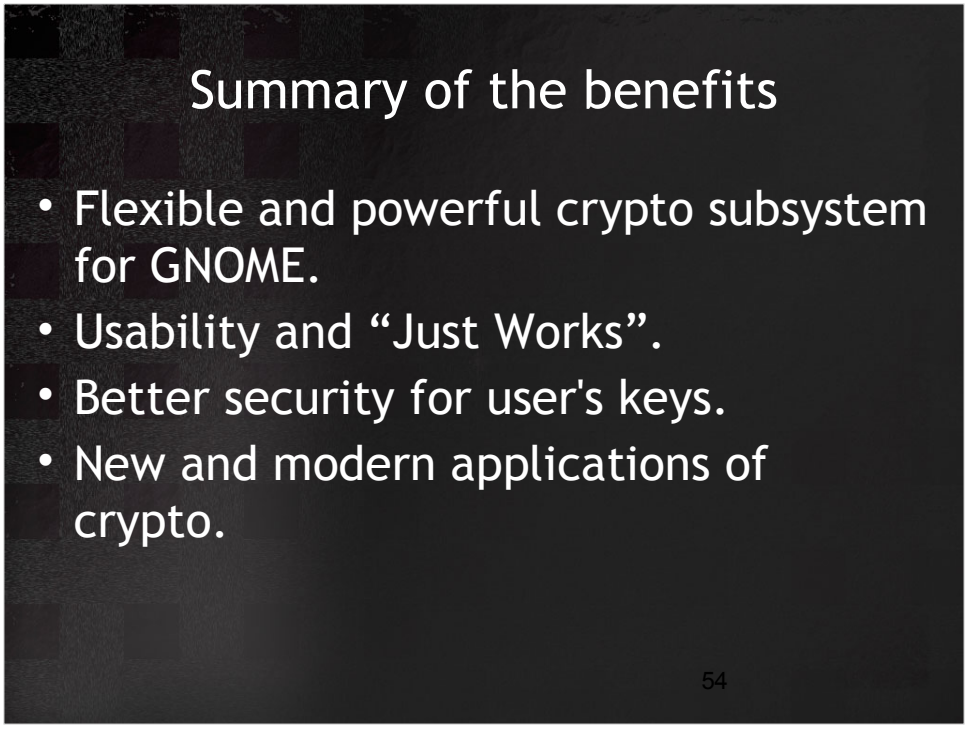


- Gooze has offered smart cards to developers working on this stuff.

What do we need to do?

1. Adjust GNOME applications to load PKCS #11 modules and look for keys and certificates there.
2. Help various libraries complete their PKCS #11 support.
3. Complete the common UI widgets.
4. Complete the GNOME Keyring PKCS #11 Key store.
5. Test and integrate Smart Cards and third party modules.

- May take some time since I've been the only one working on this stuff in GNOME.
- Want to have more help!
- Would love to be hired to work on this.
- We've done our job well when users don't notice.



Central key store can be unlocked on login or unlocking the screensaver.

Your keys and certificates are stored encrypted, unlocked when you log into your desktop.

Applications that can use smart cards without being logged in.

A pluggable and configurable way to arrange things.

Can be as secure or as usable as desired.

Questions...

http://live.gnome.org/GnomeKeyring

Credits

- Graphics: iconka.com, clker.com, HIT Entertainment
- Thanks to Reviewers.