Stef Walter.

Interested in making crypto usable.

I work for Collabora.

Maintainer of gnome-keyring and seahorse developer.

Several of us have been on a push to integrate the various crypto libraries and other excellent projects like OpenSC into the Desktop.

We've used PKCS#11 as the glue.

I've bene working with guys like Nikos to iron out the issues that come up when using PKCS#11 as glue.

From a usability perspective PKCS#11 provides applications with a way to share keys and certificates and provide a consistent and predictable behavior when using certificates.

There's PKCS#11 support in gnutls and NSS.

There's PKCS#11 support in glib, gnome-keyring, seahorse and a bunch of other projects.

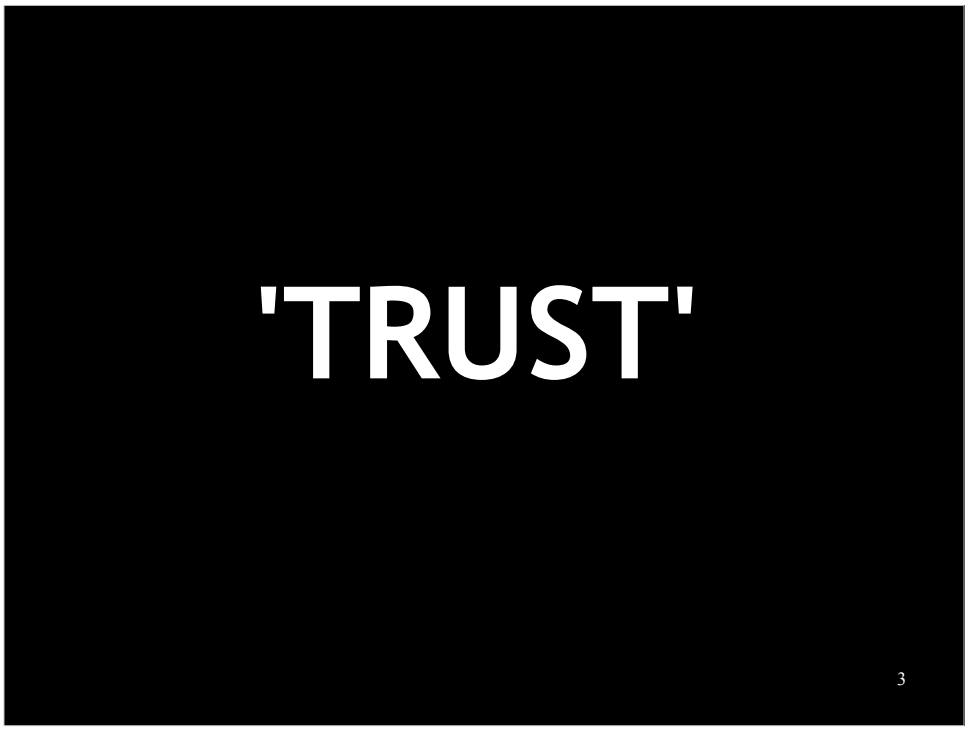
Celebrating diversity.

Gnome keyring is built around PKCS#11

Approaching the point in time when keys on your smart card (used via OpenSC) can be used by almost any application to connect to a TLS based service.

Part of the glue problem: Need consistent trust decisions.

We've worked out one way to solve this problem with PKCS#11



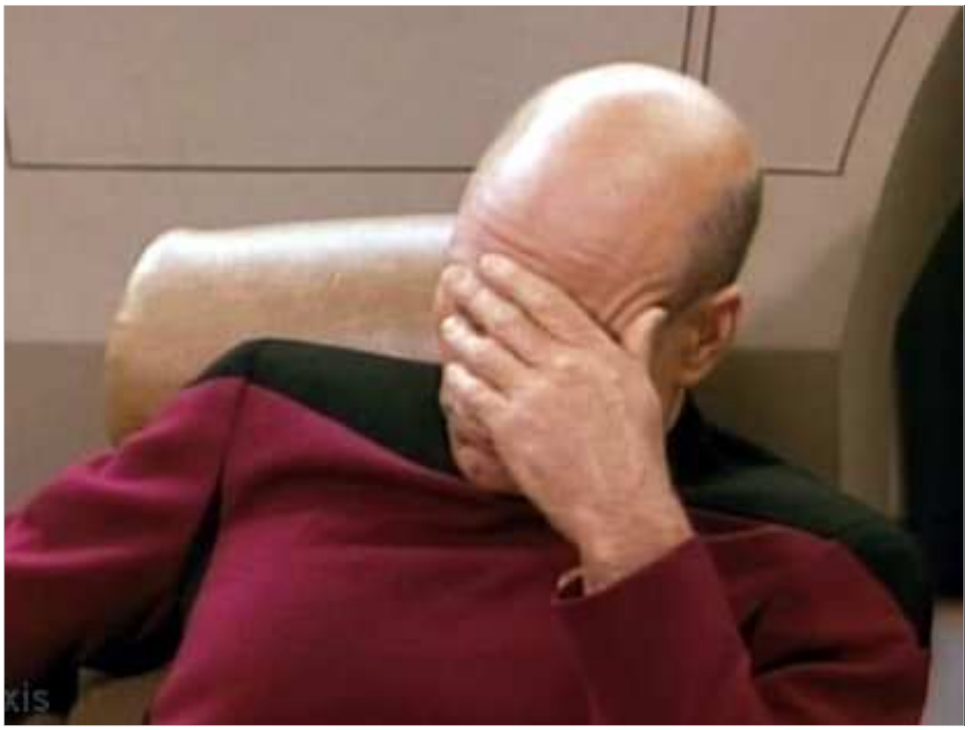
First a word about a word. 'Trust' is very ambiguous.

In security we need precise definitions, and 'trust' is precisely imprecise.

If you shout out the word 'trust' to a room of people it'll probably mean something different to everyone.

We shouldn't expose it in user interfaces, we shouldn't abuse this word.

Sadly I'm going to abuse this word.



I'm going to abuse it enough to make Jean Luc Picard do this.

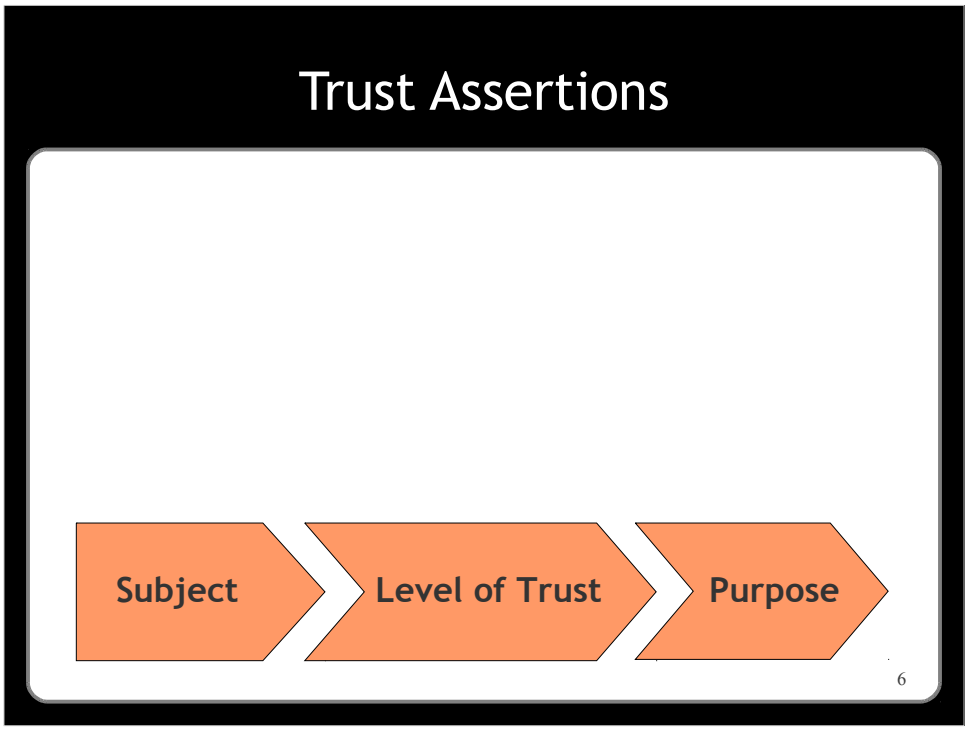
However I'd like to introduce to you a specific concept called 'trust assertions'.

A trust assertion stores a single piece of information about the user's trust in a subject for a given purpose.

Trust assertions are what we need to expose to various applications and libraries so that they can make predictable and consistent security decisions.

Trust assertions don't represent the decisions themselves but are facts that can be used when making trust decisions.

Doesn't mean that all applications have to make exactly the same trust decisions.



A trust assertion is a tuple containing: a subject, a level of trust, and a purpose.

Subject are things like a certificate, key, fingerprint or signature.

Level of trust: distrust, positive trust, introducer.

All trust has to be for a given purpose.

You trust your bank with your money, and your school with your kids, but not necessarily vice versa.

Example of a trust assertion.

A positive trust assertion.

Certificate is the subject.

Level of trust is introducer

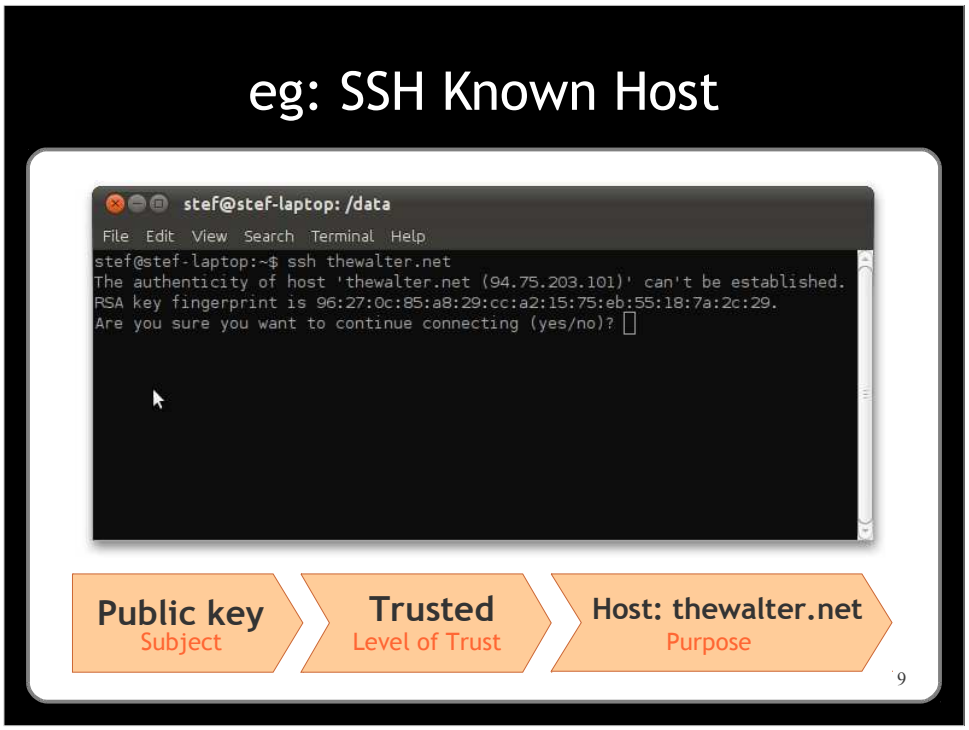
Purpose is server authentication.

Another example

A positive trust assertion.

A pinned certificate exception, used with a specific host (or peer).

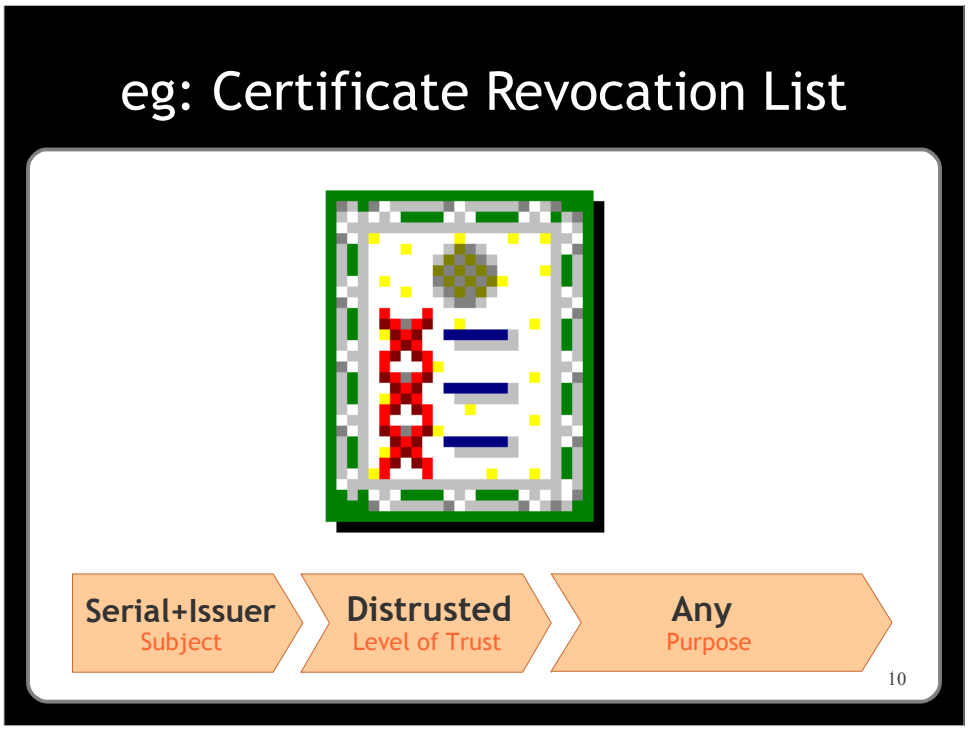
Level of trust is: Trusted.



Another example

Positive trust assertion

Subject is a public key.

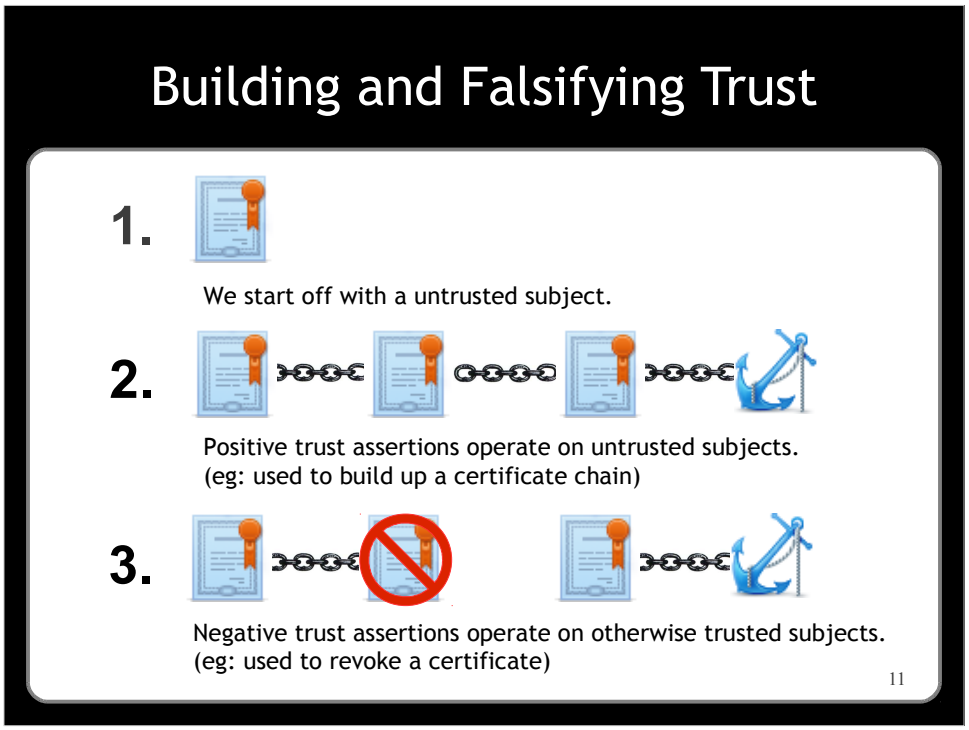


This time a negative trust assertion.

The subject is different.

The purpose is: any

Level of trust: distrusted.



How positive trust and negative trust assertions are usually used.

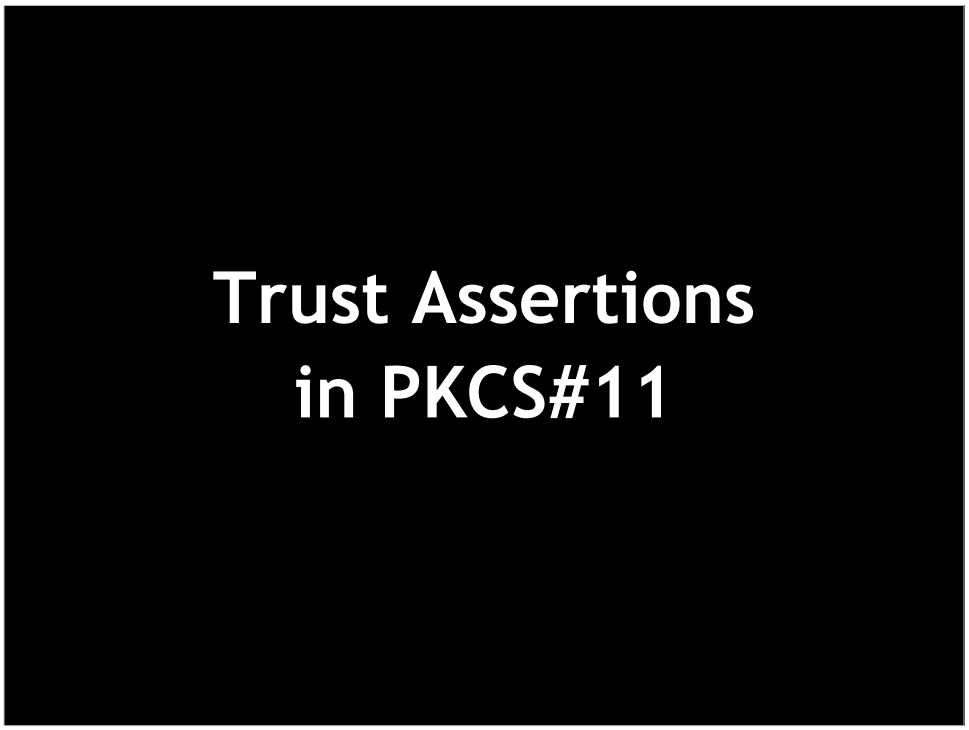
Start with untrusted (no known trust).

Use positive trust assertions to build up trust.

Negative trust assertions then falsify and tear down trust.

Positive trust assertions operate on untrusted stuff.

Negative trust assertions operate on trusted stuff, and mark stuff distrusted



We can use PKCS#11 to store trust assertions.

Trust assertions are stored in PKCS#11 objects, one object per trust assertion.

Each object has a class, and then a type, similar to public key.

The various types of trust assertions in PKCS#11 is open ended and we can define new one

Can be stored on a different token than the subject.

So this can work with any PKCS#11 provider, as long as there's one around that can store trust assertions.

PKCS#11 Assertion Objects

Anchored Certificate

Positive Trust Assertion

ATTRIBUTE TYPE	ATTRIBUTE VALUE
CKA_CLASS	CKO_X_TRUST_ASSERTION
CKA_X_ASSERTION_TYPE	**CKT_X_ANCHORED_CERTIFICATE**
CKA_X_PURPOSE	String of purpose (or OID?)
CKA_X_CERTIFICATE_VALUE	DER encoding of certificate

13

A Trust Assertion describing a certificate anchor.

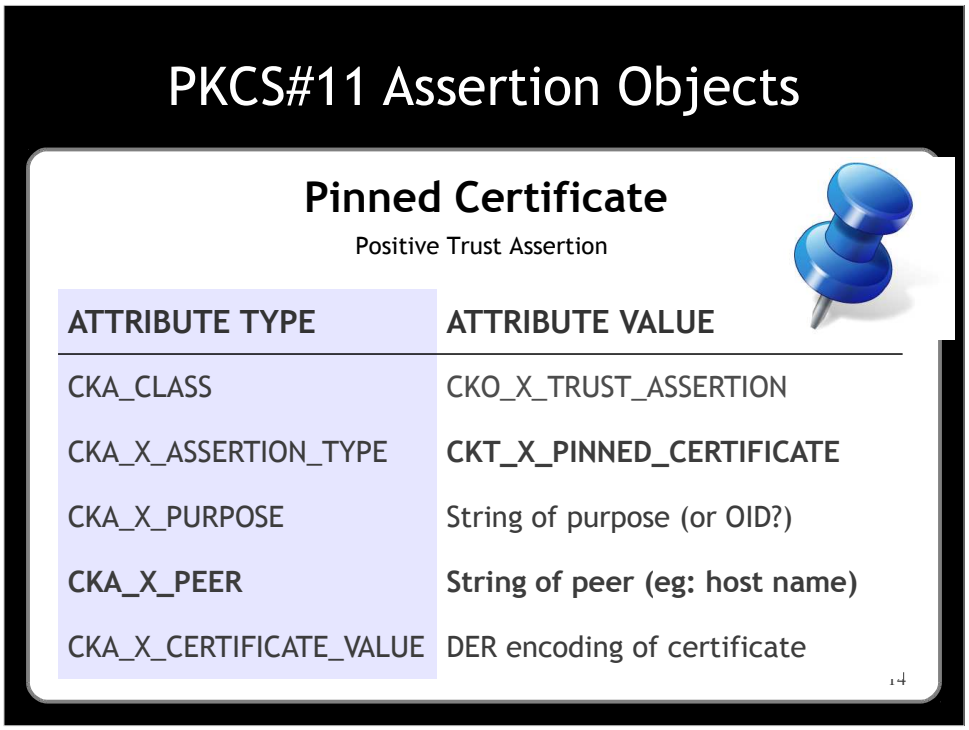
Subject is the full certificate value.

We don't use hashes for the subject, since there's no stable hash algorithm.

Using full certificate value mirrors the way things are currently done (eg: in /etc/ssl/certs)

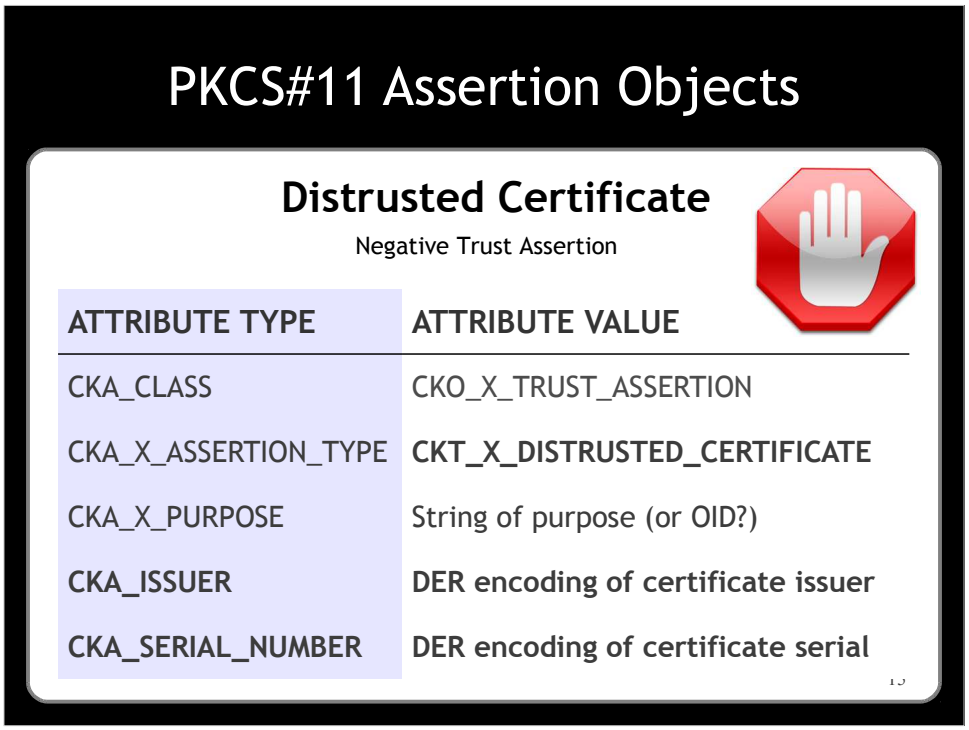
We can use any string as the purpose. Obviously we have to standardize for trust assertions that multiple apps use.

For things like server auth, client auth, email protection, code signing, we've chosen to standardize on the Extended key usage OIDs.



Pinned certificate trust assertion

Also has peer.



Example of a negative trust assertion.

A token that exposes CRLs would expose these sorts of objects.

Can't use full certificate value as subject. Have to use serial and issuer.

Implementations

Implemented

- glib
- gnome-keyring
- libgcr

Looking forward to:

- gnutls

Compatible with:

- **NSS**

Spec: http://people.collabora.co.uk/~stefw/trust-assertions.html
Short: http://goo.gl/QeR52

Some implementations of trust assertions.

We have a compat module for NSS that exposes trust assertions as trust objects.

More glue: p11-kit

- **Multiple users of a PKCS#11 module in the same process run into each other on initialize and finalize.**
- **p11-kit fixes this by providing a proxy library that coordinates and ref counts module loads.**

- **We need a configuration that multiple applications can use to load modules consistently.**
- **p11-kit provides this as library functions.**

Glue coming to a desktop near you.

Discussion of more glue work in progress. Not finalized, brainstorming going on.

Questions...

http://live.gnome.org/GnomeKeyring

Credits

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