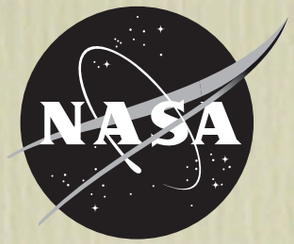
Kerberized Certificate Issuance Protocol (KX509)

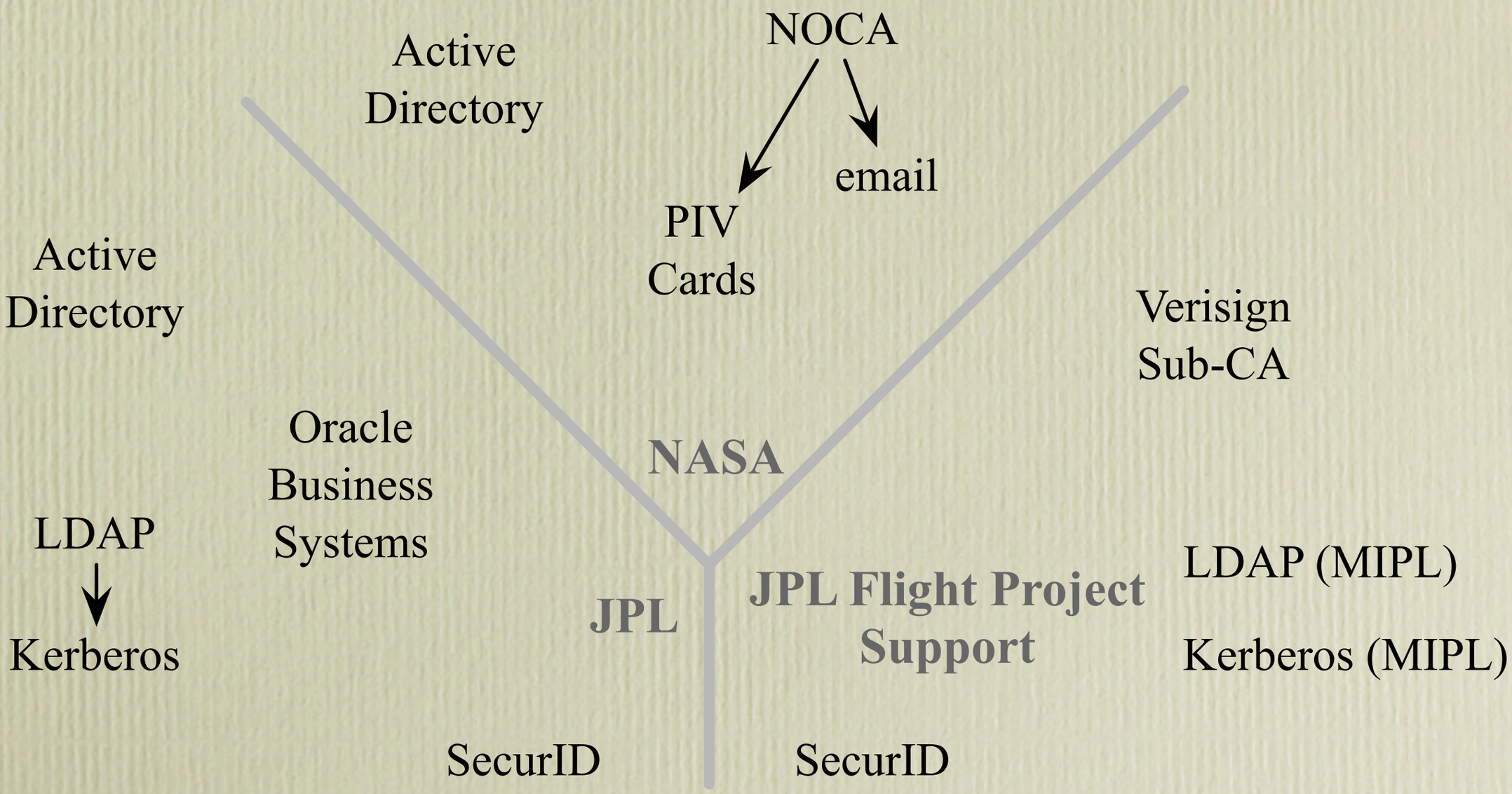
Henry B. Hotz
Jet Propulsion Laboratory



Overview and Purpose

- KX509 is a wire protocol for using Kerberos tickets to acquire X.509 certificates.
  - Kind of the opposite of PKINIT
- Where both X.509 and Kerberos are in use, want to guarantee they both authoritatively refer to the same entities.
- Already in use at several large institutions.

Identity Infrastructures at JPL



Protocol Overview

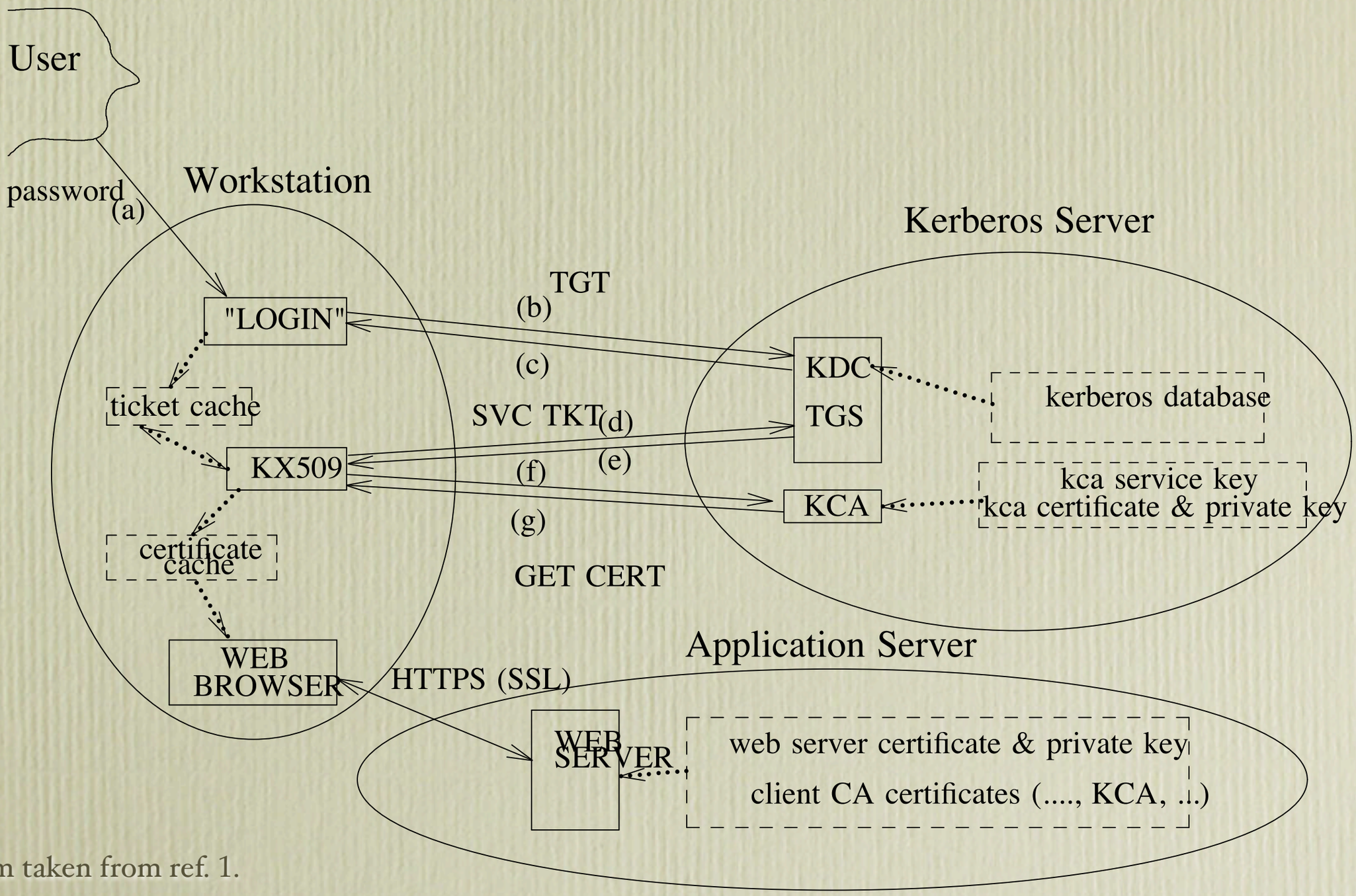


Diagram taken from ref. 1.

Client Implementation

- Command line utilities
  - `kx509`
    - Generates pub/private keys
    - Does protocol exchange with KCA
    - Stores certificate in Kerberos credential cache
  - `kxlist`
    - List certificates stored in the Kerberos credential cache
- PKCS-11 library
  - Implements PKI support using the cert/key in the credential cache.
- Interest in having the library get the cert when opened.

Protocol Description – Request

- UDP packet containing
  - version string (2.0)
  - `ap-req` – normal Kerberos stuff.
  - `pk-hash` – HMAC/SHA-1 of the version string and the `pk-key`.
  - `pk-key` – RSA public key.
    - UMICH implementation supports DSA keys, but not used “in the wild”.
- Nothing is encrypted

Protocol Description – Reply

- UDP packet containing
  - version string (`2.0`)
  - `error-code` – `0` (absent) means OK.
  - `hash` – HMAC/SHA-1 of the reply fields present.
  - `certificate` – X.509 certificate.
  - `e-text` – error message.

Reply Constraints

- All fields are nominally optional, but only the following combinations are allowed:

certificate			hash
	error-code	e-text	hash
	error-code	e-text	

- The certificate should contain:
  - Subject name unique to requestor.
  - Unique serial number (across all KCA's).
  - An extension identifying the original Kerberos identity
    - `id-pkinit-san` – preferred
    - `kcaAuthRealm` – realm only
    - `userPrincipalName` – similar to `id-pkinit-san`.

Observed Deployment

```
Version: 3 (0x2)
Serial Number: 30358893 (0x1cf3d6d)
Signature Algorithm: sha1WithRSAEncryption
Issuer: DC=gov, DC=fnal, O=Fermilab, OU=Certificate Authorities, CN=Kerberized CA HSM
Validity
    Not Before: Sep 23 18:48:37 2010 GMT
    Not After : Oct  1 15:10:31 2010 GMT
Subject: DC=gov, DC=fnal, O=Fermilab, OU=People, CN=Matt Crawford, CN=UID:crawdad
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    RSA Public Key: (1024 bit)
        Modulus (1024 bit):  . . .
        Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Basic Constraints: critical
    CA:FALSE
X509v3 Key Usage: critical
    Digital Signature, Key Encipherment
Netscape Cert Type:
    SSL Client
Netscape Comment:
    Certificate issued by Fermilab KCA
X509v3 Issuer Alternative Name:
    email:nightwatch@fnal.gov
Netscape CA Policy Url:
    URL:http://security.fnal.gov/policies/pki_policy_certification_practices.htm
X509v3 Authority Key Identifier:
    keyid:EB:A3:7E:89:49:47:82:DA:76:C1:AA:8F:33:65:09:A2:A7:37:BF:7C
X509v3 Subject Key Identifier:
    D8:FB:1A:02:D0:63:59:5E:B5:BC:AC:08:96:DF:2B:34:12:42:0C:96
X509v3 Certificate Policies:
    Policy: 1.3.6.1.4.14147.1.8.1
      CPS: http://security.fnal.gov/policies/pki_policy_certification_practices.htm
X509v3 CRL Distribution Points:
    URI:http://security.fnal.gov/pki/99f9f5a3.0
KCA Authentication Realm:
    ..FNAL.GOV
X509v3 Subject Alternative Name:
    email:crawdad@fnal.gov, othername:<unsupported>
```

Unsupported othername is actually an `id-pkinit-san`.

Security Issues

- All Kerberos and X.509 considerations still apply.
- Don't do PKINIT with a KX509-issued cert
  - ...unless you really know what you're doing.
    - Yes, I know loops could be fun. Might even be useful occasionally.
- Understand how Kerberos and PKI policies relate.
  - Ticket/cert lifetimes.
  - Auditing headaches getting a publicly-recognized KCA.

Fundamental Limitations

- Everything is in the clear.
  - Hash should protect everything's integrity.
  - Privacy/anonymity is *not* supported.
- Public key can be sniffed and reused.
  - Requestor does not have to prove knowledge of the secret key.
  - Breaks non-repudiation/digital signature applications.
    - Don't deploy with those Key Usage bits.
  - Any usage should prove knowledge of the secret key, independent of the cert.
    - TLS client OK.

Future (Version 3.0?)

- This version has a lot of warts, mostly due to its age.
  - Originally developed with Kerberos 4!
  - Following suggestions made on IETF lists.
- Replace components of exchange with current standards which provide equivalent functionality.
  - Use Kerberos checksum instead of SHA1-HMAC
    - Probably should use KRB_XXXX or GSSAPI packaging.
  - Send PKCS-10 (RFC-2986) signed request instead of bare public key.
    - Request should also tie to Kerberos identity.
  - e-text should be UTF8, not VisibleString

Other Enhancements

- Use TCP instead of UDP
- Define a new message that says “wait”.
  - Allow for external attribute lookup operations or other complications.
- Return the entire cert chain, not just the new end-entity cert.
- Add an identifying type extension for issued certificates.

References

- Doster, W., Watts, M., and D. Hyde, "The KX509 Protocol", September 2001, <http://www.citi.umich.edu/techreports/reports/citi-tr-01-2.pdf>
- draft-hotz-kx509-01 <http://tools.ietf.org/>
  - Thanks to all the people acknowledged in Section 4 that draft.
- IETF Kerberos and PKIX mailing lists.