Centralised User Management

What's AAA?

- Authentication
  - "Who are you?"
- Authorisation
  - "What are you allowed to do?"
- Accounting
  - "What did you do?"

Centralised

- Because we need control of the systems we own and manage
- We need scalability in management
    - e.g. not have to reconfigure hundreds of machines every time someone joins or leaves
- We need something which is auditable - confidence that we haven't accidentally missed something

Solution presented here

- KERBEROS for authentication, LDAP for authorisation (and SYSLOG for accounting)
- We'll be using open source: MIT Kerberos and OpenLDAP
- Microsoft Active Directory is basically Kerberos + LDAP + DNS; if you like it, by all means use it
    - Microsoft's tweaked versions of protocols
    - May require extra configuration, e.g. install Microsoft Services For Unix (SFU)

Kerberos overview

- Based on symmetric (private key) cryptography
  - Secure, fast, scalable
- Provides **true single sign-on**
  - Type your password once at start of day
  - Your password is never sent to services you use!
- KDC: Key Distribution Centre
- REALM: Collection of users and machines which all trust the same KDC. Named in UPPER.CASE to distinguish from a DNS domain

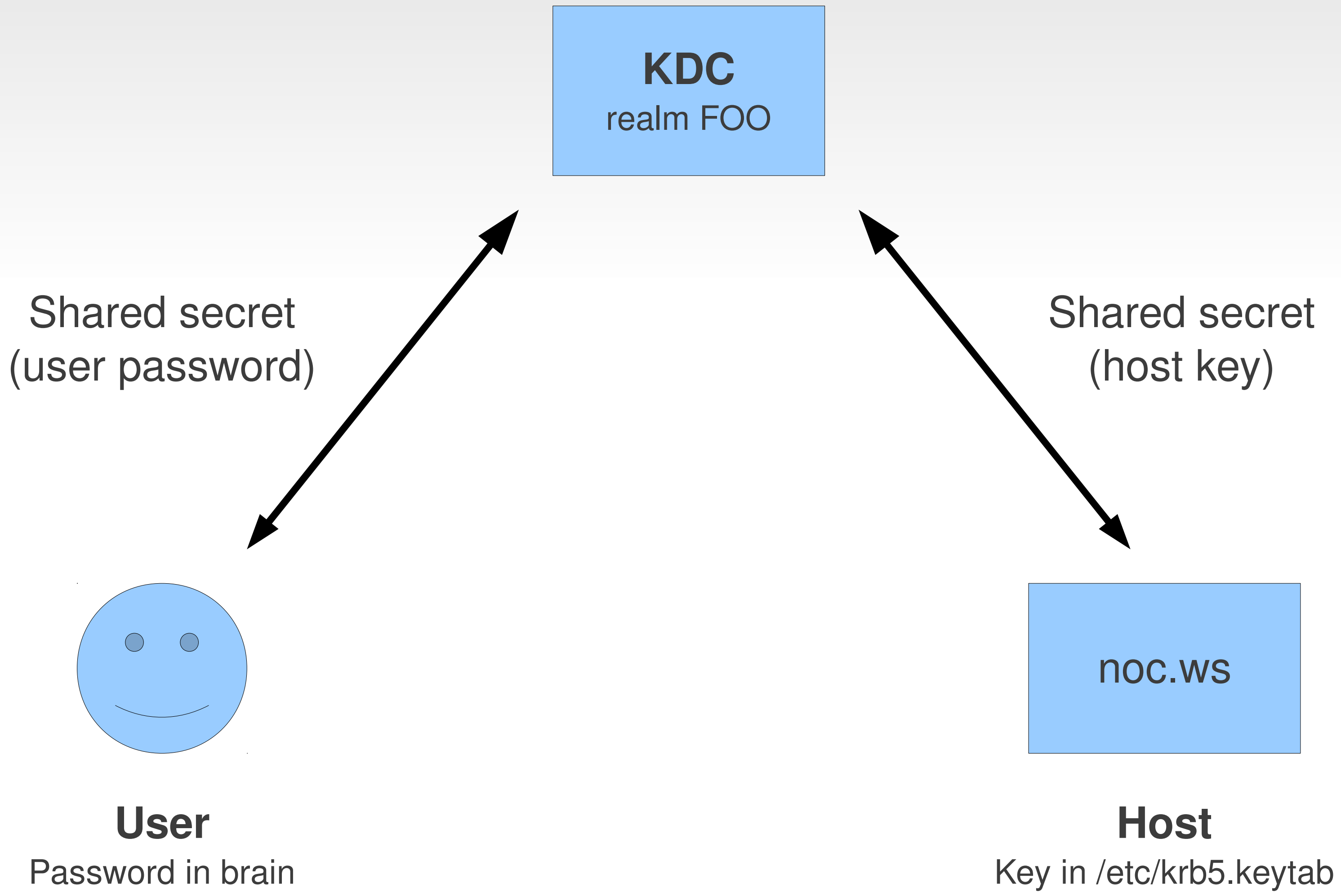
Informal protocol design: http://web.mit.edu/kerberos/www/dialogue.html

KDC Database

- A simple table of "things" and "passwords"
  - users:
    - myname@WS.NSRC.ORG
  - hosts:
    - host/noc.ws.nsrc.org@WS.NSRC.ORG
  - services:
    - HTTP/www.ws.nsrc.org@WS.NSRC.ORG
  - Kerberos calls them all "principals"
  - Passwords (shared secrets) stored in clear text
    - Actually, entered password munged into a binary key

Shared secrets

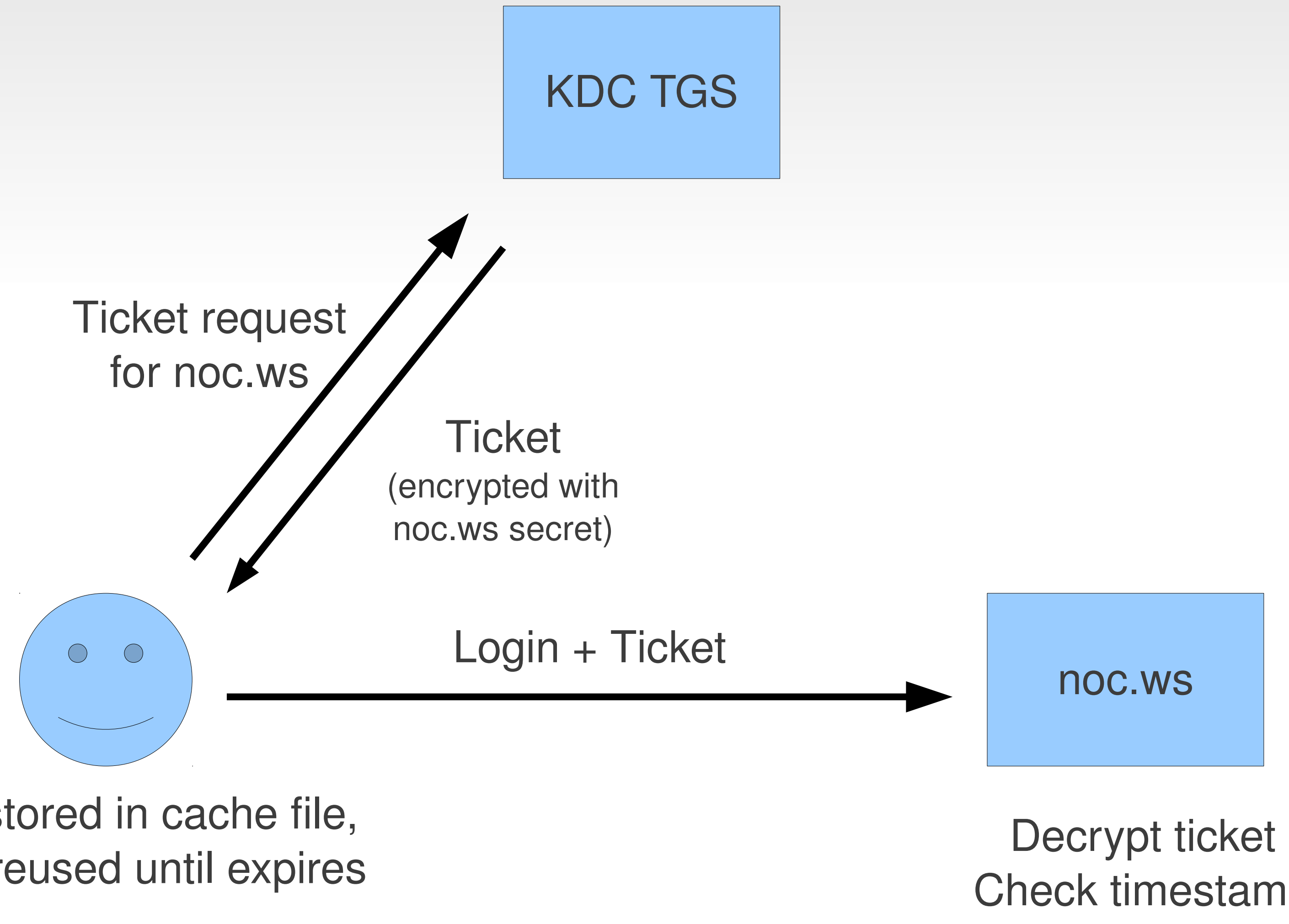
principal	key
myname@FOO	XXXXX
host/noc.ws@FOO	YYYYY



Prove identity via "tickets"

- When you want to access a service, you first obtain a "ticket" for that service
- The KDC sends you the ticket, which is encrypted with the service's key
    - Only the KDC and the service know this key
    - Hence the service knows that the ticket must have come from the KDC
- You never send your password to the service
- Tickets are time-limited (typically 10 hours)

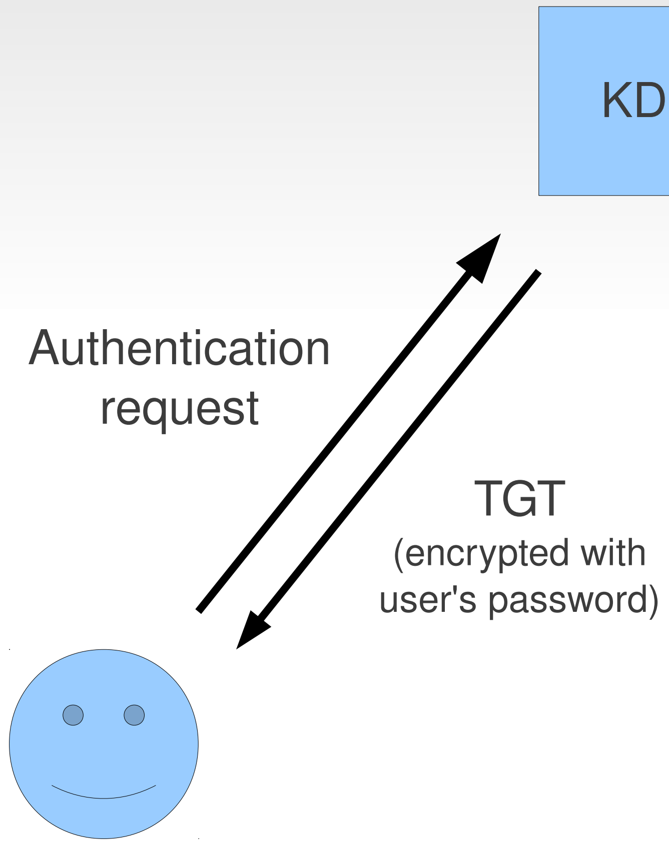
Ticket Granting Service



Obtaining tickets

- Each ticket is only readable by one particular host/service, so you need to obtain a ticket for each one
- To avoid having to enter your password each time, you first obtain a master ticket: a "ticket granting ticket"
- Your TGT is encrypted with your own password, and decrypted when you receive it
- Unix program: **kinit**
  - See also: **klist**, **kdestroy**

Authentication Service



kinit
Decrypted TGT stored in cache file

Future-proof:
The AS can be replaced with another auth mechanism (e.g. smartcards), but the TGT and tickets are the same

Practical: Kerberos client

```
sudo apt-get install krb5-user
sudo editor /etc/krb5.conf

# delete everything, replace with this:
[libdefaults]
default_realm = WS.NSRC.ORG
dns_lookup_realm = true
dns_lookup_kdc = true
```

Magic which I will explain later

```
sudo editor /etc/ssh/ssh_config # OSX: /etc/ssh_config
  ...
  GSSAPIAuthentication yes        # check this line
  GSSAPIKeyExchange yes           # optional extra
  ...

kinit testuser
# enter password when prompted
ssh testuser@noc.ws.nsrg.org
# logout, then login somewhere else
ssh testuser@s1.ws.nsrc.org
```

Simple enough?

- Easy to train your users - they don't need to know how it works
- Little work to configure client machines (scalable)
- Of course, we had to build the server side first :-)

Multi-protocol support

- What we've seen so far is similar to what you can do with ssh + pubkey authentication + ssh-agent (*)
- But Kerberos can authenticate many other protocols: POP3, IMAP, HTTP, LDAP, even telnet!
  - Bolt-on using SASL and GSSAPI
- Also optionally adds encryption to those protocols
- Oh, and it does mutual authentication too
  - No need to have CA certificate or ssh host keys
    (For ssh, set "GSSAPIKeyExchange yes" on server and client)
    http://www.sxw.org.uk/computing/patches/openssh.html

(*) You can even forward your kerberos tickets to another host, like ssh agent forwarding

Demo: HTTP with Kerberos

```
http://noc.ws.nsrc.org/secure

You need to configure your client to attempt Kerberos
authentication.

#### For curl ####
curl --negotiate -u : http://......

#### For Firefox ####
Go to about:config
Filter on "negotiate"
network.negotiate-auth.trusted-uris   ws.nsrc.org

#### For Google Chrome ####
Start using:
/opt/google/chrome/google-chrome \
  --auth-server-whitelist=*.ws.nsrc.org

# Use kdestroy and kinit to convince yourself!
```

Demo: LDAP with Kerberos

```
sudo apt-get install ldap-utils
sudo apt-get install libsasl2-modules-gssapi-mit

ldapsearch -Y GSSAPI -H ldap://ldap.ws.nsrc.org \
  -b "dc=ws,dc=nsrc,dc=org" "(cn=*test*)"

# Note:
# (1) No password prompt! (kdestroy to confirm)
# (2) Data encrypted (tcpdump to confirm)
```

Kerberos and DNS

Locating KDCs for realm

- Client needs to locate the KDC(s) for a realm
- This can be statically configured in krb5.conf

  -
    ```
    [realms]
      WS.NSRC.ORG = {
            kdc = kdc1.ws.nsrc.org
            kdc = kdc2.ws.nsrc.org
            admin_server = kdc1.ws.nsrc.org
      }
    ```
- Or we can lookup SRV records in DNS
- This saves configuration on the clients

```
$ dig _kerberos._udp.ws.nsrc.org srv
;; ANSWER SECTION:
_kerberos._udp.ws.nsrc.org. 600 IN SRV 0 100 88 kdc1.ws.nsrc.org.
_kerberos._udp.ws.nsrc.org. 600 IN SRV 0 100 88 kdc2.ws.nsrc.org.
```

Host to realm mapping

- When you connect to a host, you need to know what realm it is in (so client can get the right ticket)
- You can configure this statically in krb5.conf

  -
    ```
    [domain_realm]
      .ws.nsrc.org = WS.NSRC.ORG
    ```
- Or again you can use the DNS

Host to realm algorithm

- Reverse lookup IP to FQDN (foo.bar.baz.com)
- Look for TXT record in turn:
  - _kerberos.bar.baz.com
  - _kerberos.baz.com
  - _kerberos.com
- Fallback is FQDN without hostname, uppercased
  - Note: the "default_realm" from krb5.conf *isn't* used

```
$ dig _kerberos.ws.nsrc.org txt
;; ANSWER SECTION:
_kerberos.ws.nsrc.org. 600 IN TXT  "WS.NSRC.ORG"
```

Importance of DNS

- It's critical that forward *and reverse* DNS is correctly configured for all servers you connect to, or it won't work
- Multi-homed hosts need care. Either:
  - one hostname, multiple A records, all PTR records point to same hostname
  - or: separate hostname for each interface, with matching forward and reverse
  - See the Kerberos FAQ for more info

Kerberos gotchas

- Clocks must be synced (within 5 minutes)
- Realm must be in UPPER.CASE
- Target must have correct forward+reverse DNS
- Target must know its own hostname
  - Check "hostname", /etc/hostname
- Not too difficult?
  - if you can get these things right, you can turn off ssh password authentication entirely
  - if it breaks, can still get in on the console to fix it