Admin Security

Tim Quinn
Cluster Infrastructure Meeting
July 13, 2010

Agenda

- Look at...
  - High-level requirements/design goals
  - Implementation approach
- ...for...
  - Admin client ↔ DAS
  - DAS ↔ instance
- Throughout, some use cases
  - Steady-state
  - Bootstrapping
- Some possible to-do items

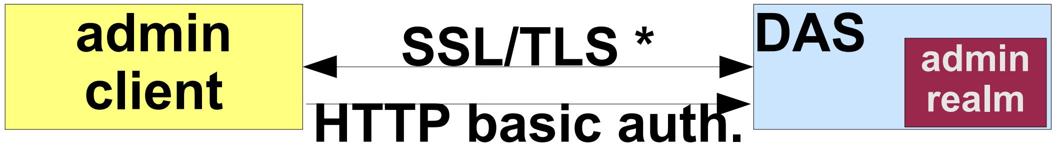
General Requirements/Design Goals

- Command-line compatibility with GlassFish 2
- Elective – admin security not required
- When elected:
  - *Never* send sensitive information in the clear
  - Secure all traffic among clients, DAS, instances

Admin client ↔ DAS High-level requirements

- As in GlassFish 2
  - User has confidence in DAS
  - DAS has confidence in user
  - Both have confidence in transport
- Different:
  - *No* sensitive data sent in cleartext

Admin client ↔ DAS Design Approach



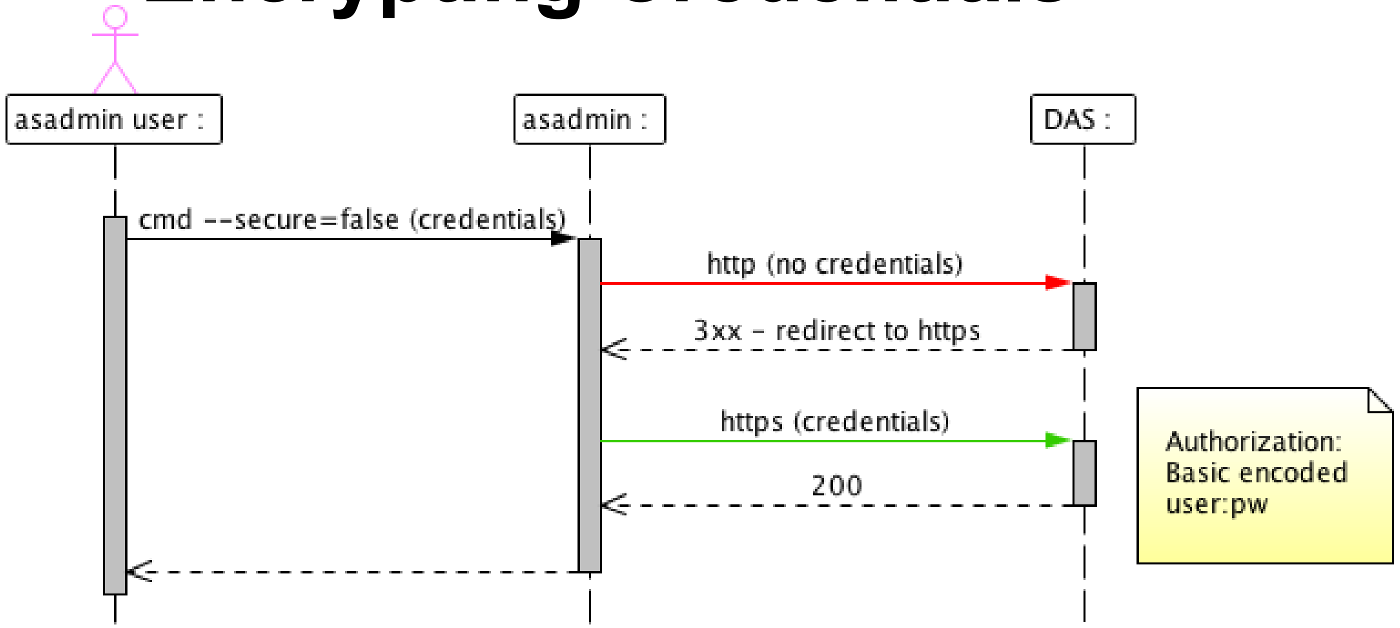
* SSL/TLS

* server auth.

  * Grizzly listener at http://das:4848 → https:/das:4848
  * Encryption: confidentiality, integrity protection
  * Authentication: DAS identifies itself to client
* HTTP
  * Header "Authorization: Basic [encoded user:password]"
* Just like GlassFish 2 ***except:***
  * Credentials (username/password) always encrypted in GF 3

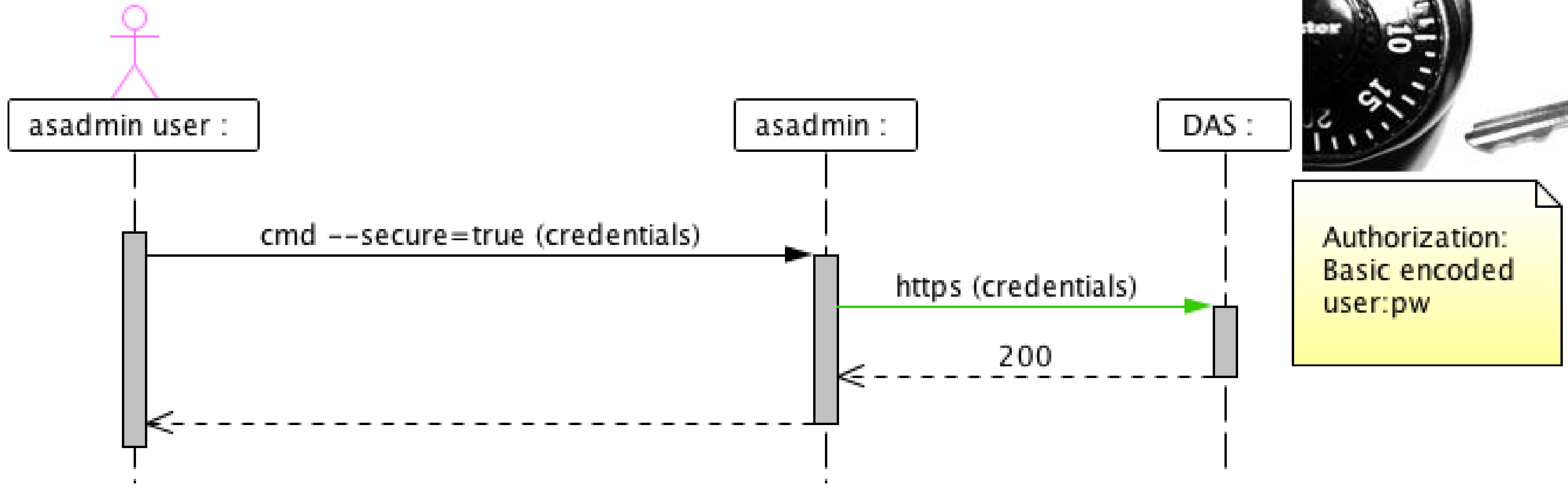
ORACLE

asadmin ↔ DAS --secure=false Encrypting Credentials



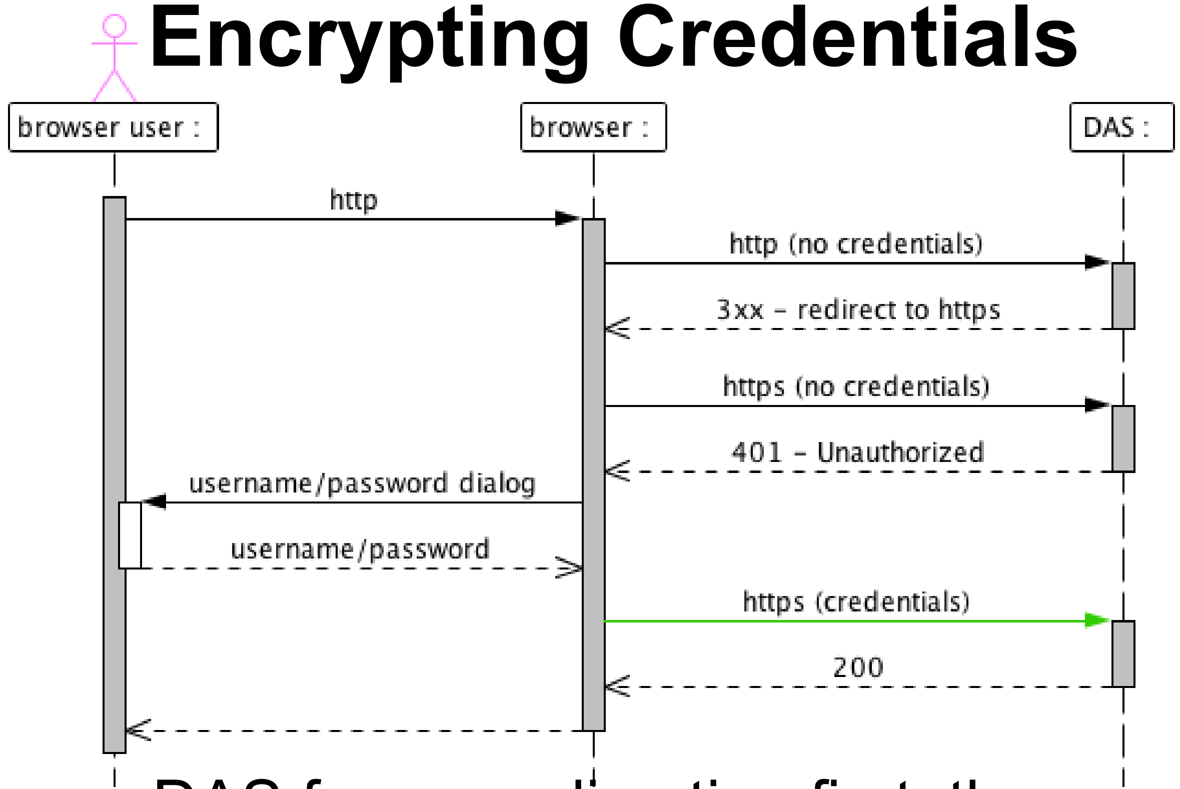
- User specifies credentials on command line
- asadmin withholds creds – connection is insecure
- DAS insists on SSL, redirects to https
- asadmin sends credentials, rest of command once connection is secure

asadmin ↔ DAS --secure=true Encrypting Credentials


asadmin user :
asadmin :
DAS :
cmd --secure=true (credentials)
https (credentials)
200
Authorization: Basic encoded user:pw

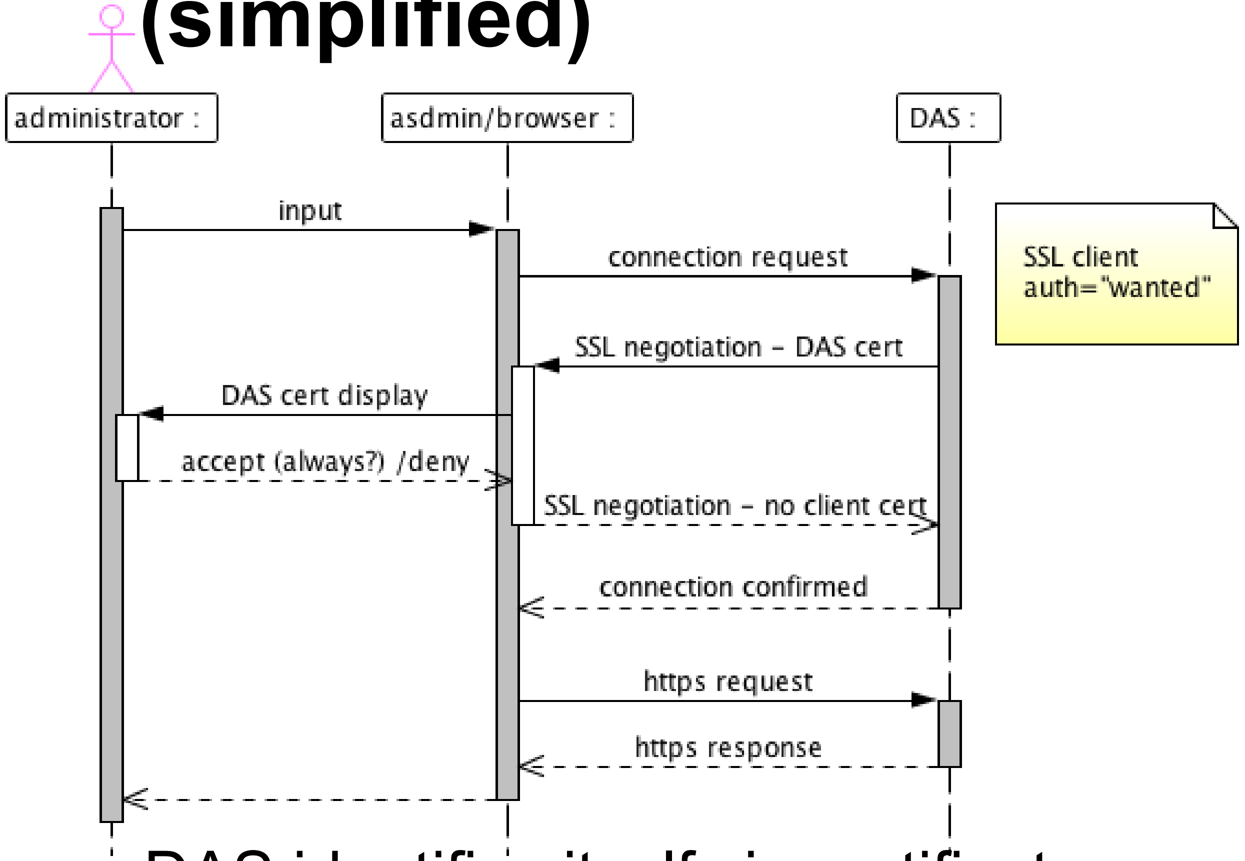
- User specifies credentials *and* secure connection
- asadmin initiates https itself, sends creds

Browser ↔ DAS Encrypting Credentials



- DAS forces redirection first, then...
- ...browser follows redirection (still no credentials)...
- ...DAS challenges for credentials
- ...browser prompts for, collects, then sends creds

A Brief Aside: SSL negotiation (simplified)

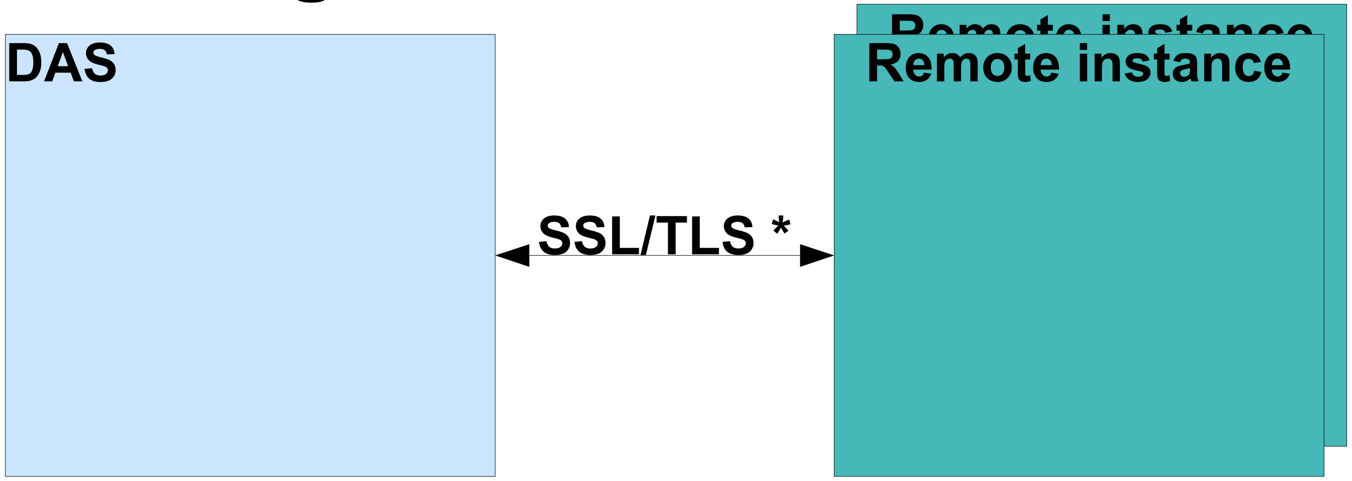


- DAS identifies itself via certificate
  - End-user accepts, perhaps “for always”
- Client *does not* typically identify using cert

DAS ↔ Instance
High-level requirements

- Secure traffic between DAS, instances
- Do not store admin password in clear
- Help prevent rogue direct connections
  - Admin client ↔ instance
  - Instance ↔ instance
  - DAS ↔ DAS

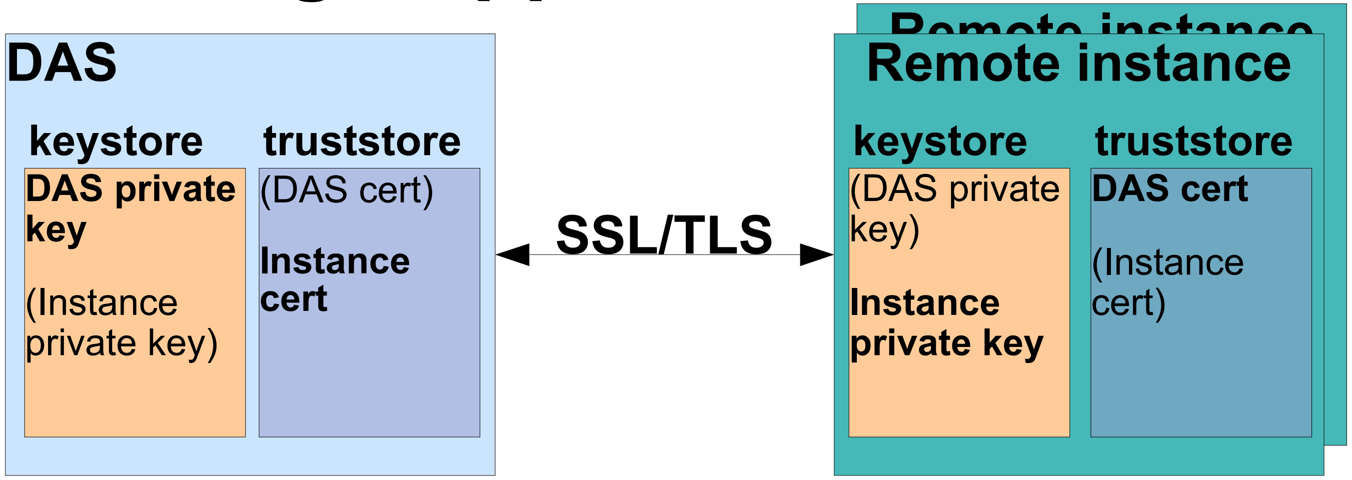
DAS ↔ Instance Design Goals



* mutual auth.

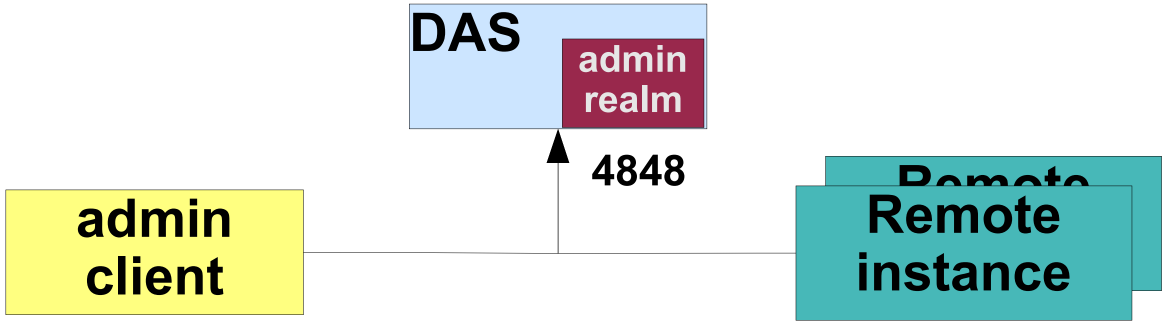
- SSL/TLS mutual authentication
- Cert-based, not username/password-based

DAS ↔ instance Design Approach



- DAS, instance use copies of same keystore, truststore
  - Avoids problems with DAS → instance sync
- DAS authenticates w/ one cert, instances use one other
- DAS trusts instance cert, instance trusts DAS cert

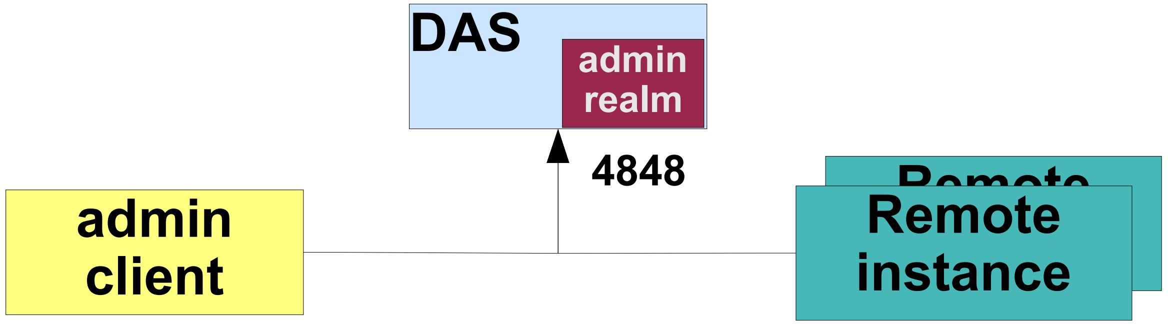
DAS ↔ Instance
One DAS admin port



Grizzly configuration

- Port unification – one port serves both http,https
- Redirection: http://das:4848 → https://das:4848
- SSL: client auth=”want” (not “need”)

DAS ↔ Instance AdminAdapter Logic



Accepts message if:

- Sending Principal in truststore and != itself

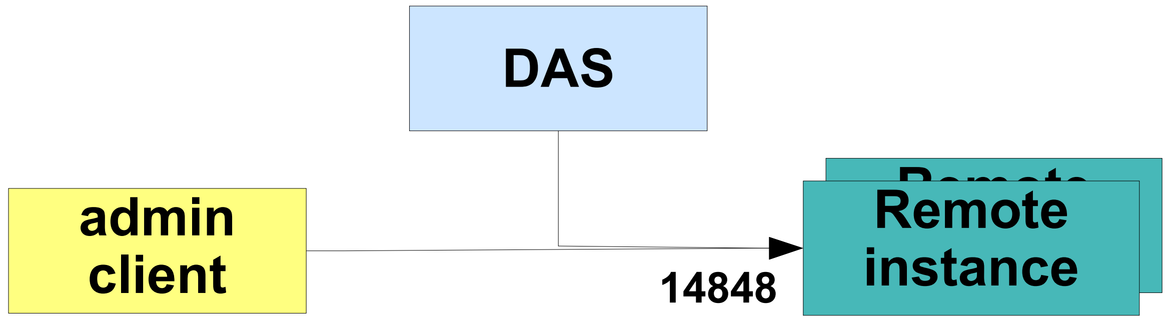
OR

- HTTP Authentication header specifies valid admin user/pw (issues challenge if header absent)

OR

- Password == provisioned local password

DAS ↔ Instance
One Instance admin port



- Grizzly configured *exactly* as on DAS
  - Uses copies of same keystore, truststore
  - Client auth=”want”
- AdminAdapter accepts message if:
  - Sending Principal != itself, **OR**
  - HTTP Authentication header specifies valid user/pw, **OR**
  - Password == provisioned local password

Authentication Summary

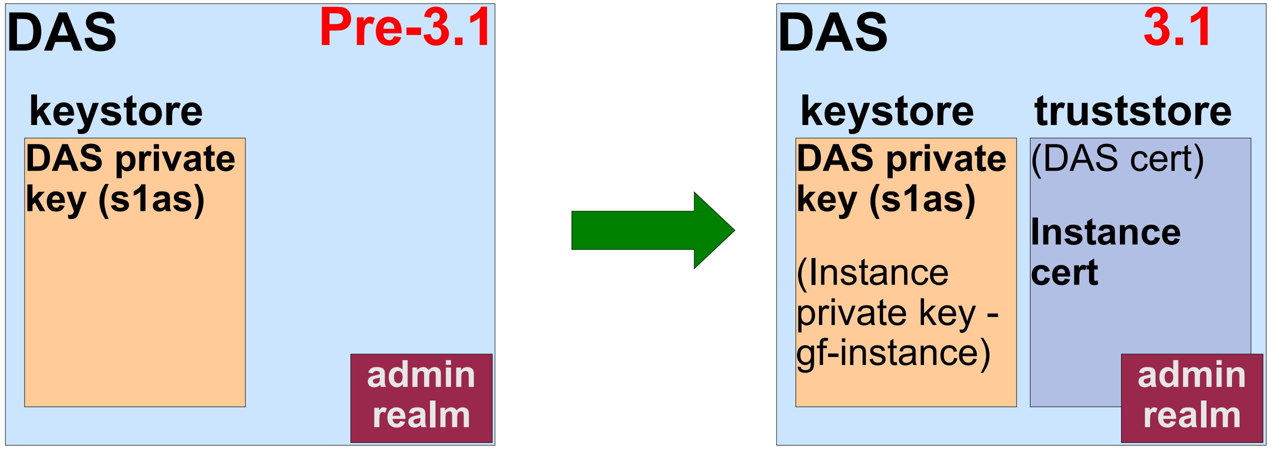
This ↓	Authenticates to		
	Client	**DAS**	**Instance**
Client	n/a	username/pw local password	local password
DAS	SSL server auth	X	SSL mutual auth
Instance	SSL server auth	SSL mutual auth	X

X = SSL permits connection, AdminAdapter rejects message (Principal == self)

Bootstrapping

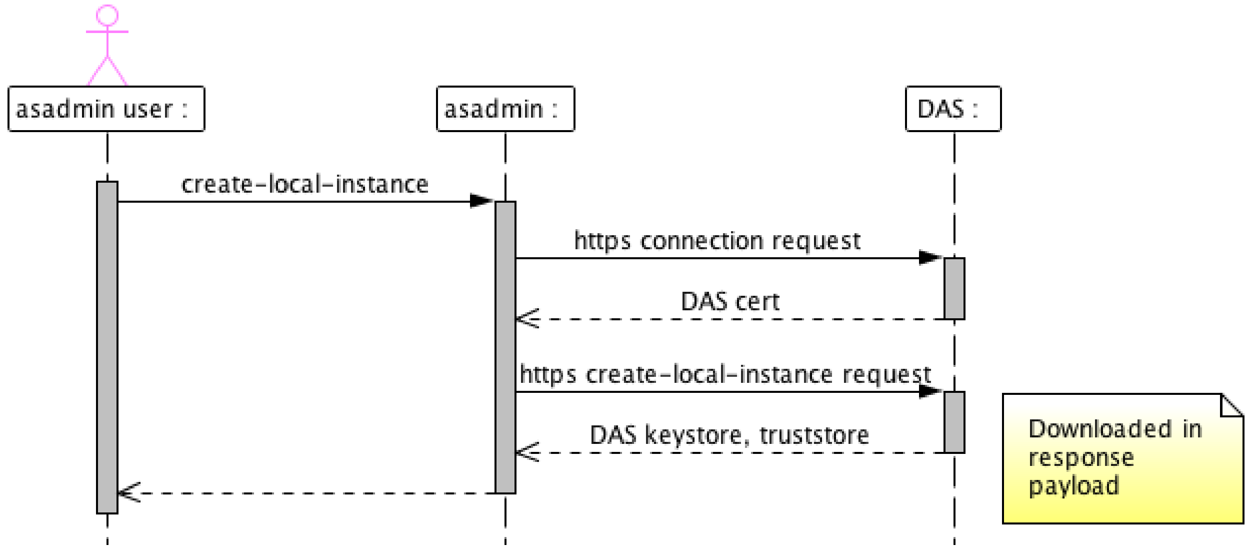
- DAS
- Create instance locally
- Create instance remotely

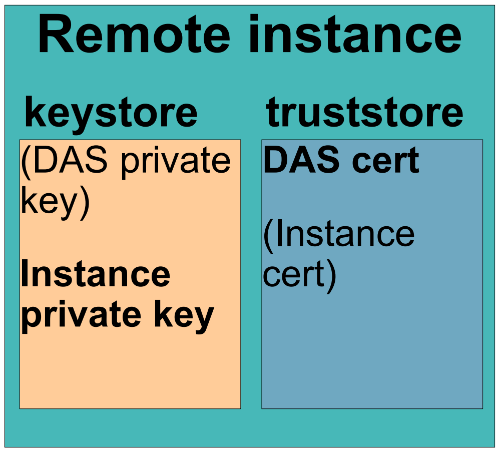
Bootstrapping DAS



- During build/create-domain:
  - Create truststore, add s1as public cert to truststore
- During initial domain start-up (or “slightly later”):
  - Generate self-signed key pair for instances to use
    - Save private key in keystore with alias gf-instance (e.g.)
    - Save public cert in truststore with alias gf-instance
    - Add gf-instance to admin realm

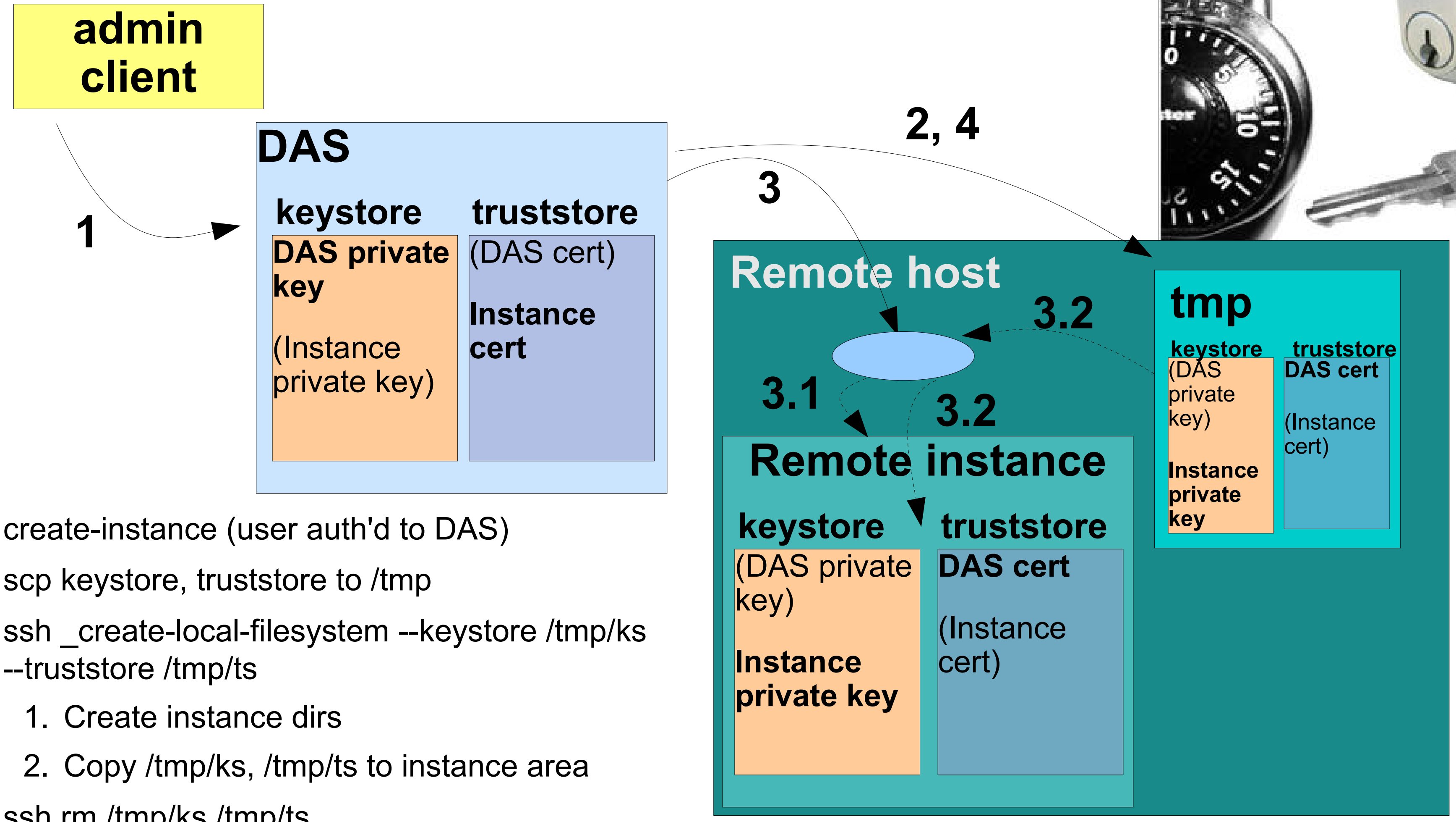
Bootstrapping
Create instance locally





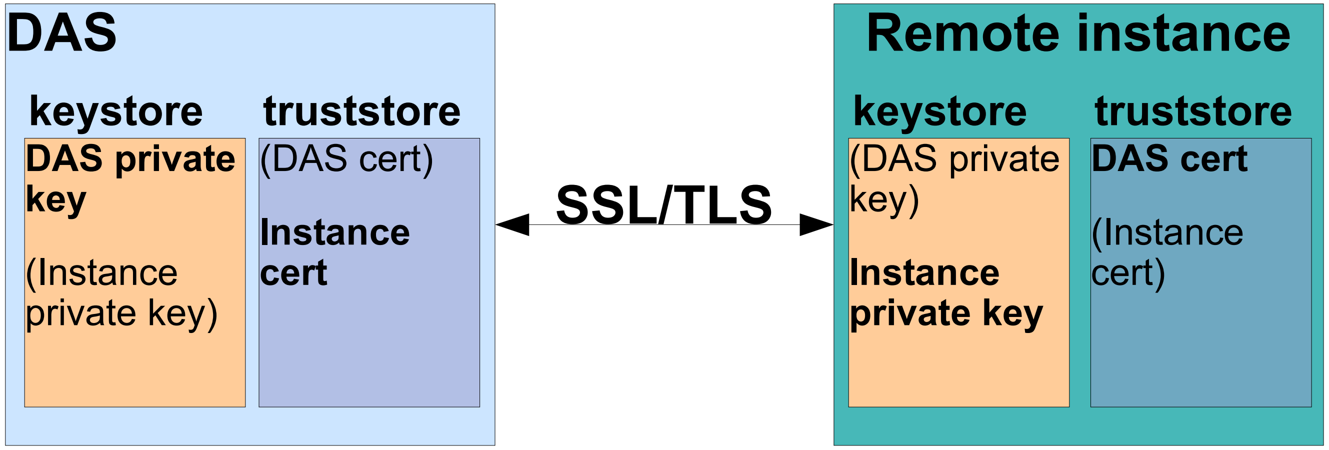
- When instance starts it has correct keystore, truststore for mutual auth with DAS

Bootstrapping
Create instance remotely



1. create-instance (user auth'd to DAS)
2. scp keystore, truststore to /tmp
3. ssh _create-local-filesystem --keystore /tmp/ks --truststore /tmp/ts
    1. Create instance dirs
    2. Copy /tmp/ks, /tmp/ts to instance area
4. ssh rm /tmp/ks /tmp/ts

Bootstrapping
Create instance locally or remotely



Whether by create-local-instance or create-instance;

- Correct keystore, truststore in place on instance
- start-instance time:
  DAS ↔ instance mutually authenticate

Some To-do Items...

Ease of use:

- Simple way for administrator to turn on, off?
- Allow administrator to update keys, certs then distribute to instances