UNICORE security stack

*P. Bala, **K. Benedyczak**, S. van den Berghe, R. Menday, B. Schuller*

Outline

- Scope of this presentation.
- Outside world: GSI security model.
- UNICORE world:
  - authentication,
  - trust delegation.
- ETD versus Proxy certificates.
- Future of UNICORE security.
  - Current problems and possible solutions.

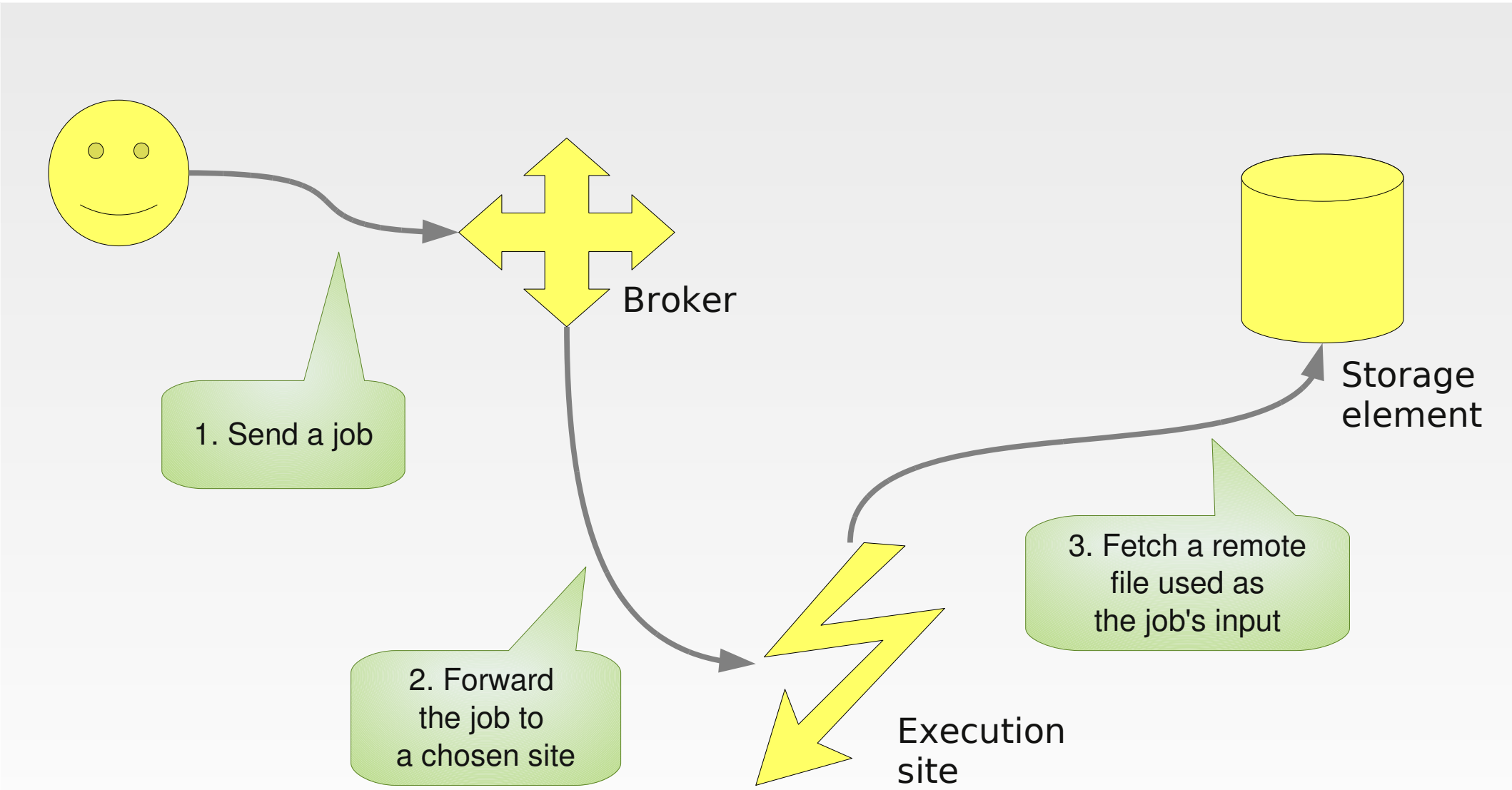
Scope of this talk

- Only low level security will be presented: authentication and trust delegation.
- Authorization, user management, VOs won't be discussed.
- Comparison with GSI model is important as it is de facto a standard in the grid world.

Authentication and trust delegation

- Authentication is the process of verification a communication peer's identity.
- Trust delegation is a process of assigning one's privileges to a trusted 3rd party.
  - *Privileges delegation* would be a definitively better name.
- A lot of distributed systems do not use trust delegation at all.
  - Simple client-server systems, where a remote service works only locally on a behalf of a user and the service is privileged to change its effective user id. Example: SSH daemon.
  - Statically configured systems, where all sites know each other. E.g. systems using `/etc/hosts.equiv`
  - Systems which take over users' credentials. For example typical Push-email provider for cell phones, requires you to give it a password to your remote mailbox.

Example of trust delegation application



Grid Security Infrastructure

- In Globus, gLite and ARC authentication, trust delegation and SSO is achieved by usage of Proxy Certificates.
- The initial proxy certificate is issued by a user.
  - Its DN is similar to the user's DN.
  - Proxy contains a new public key and is accompanied by a corresponding private key.
- When user tries to use a remote service, middleware uses the initial proxy to issue another proxy – for the service.
- The generated proxy is stored in the FS and private key is never encrypted.
- The service use the proxy to initiate a SSL/TLS connection
- It is used for SSO on the user's machine as globus/gLite/... are composed of hundreds of programs.

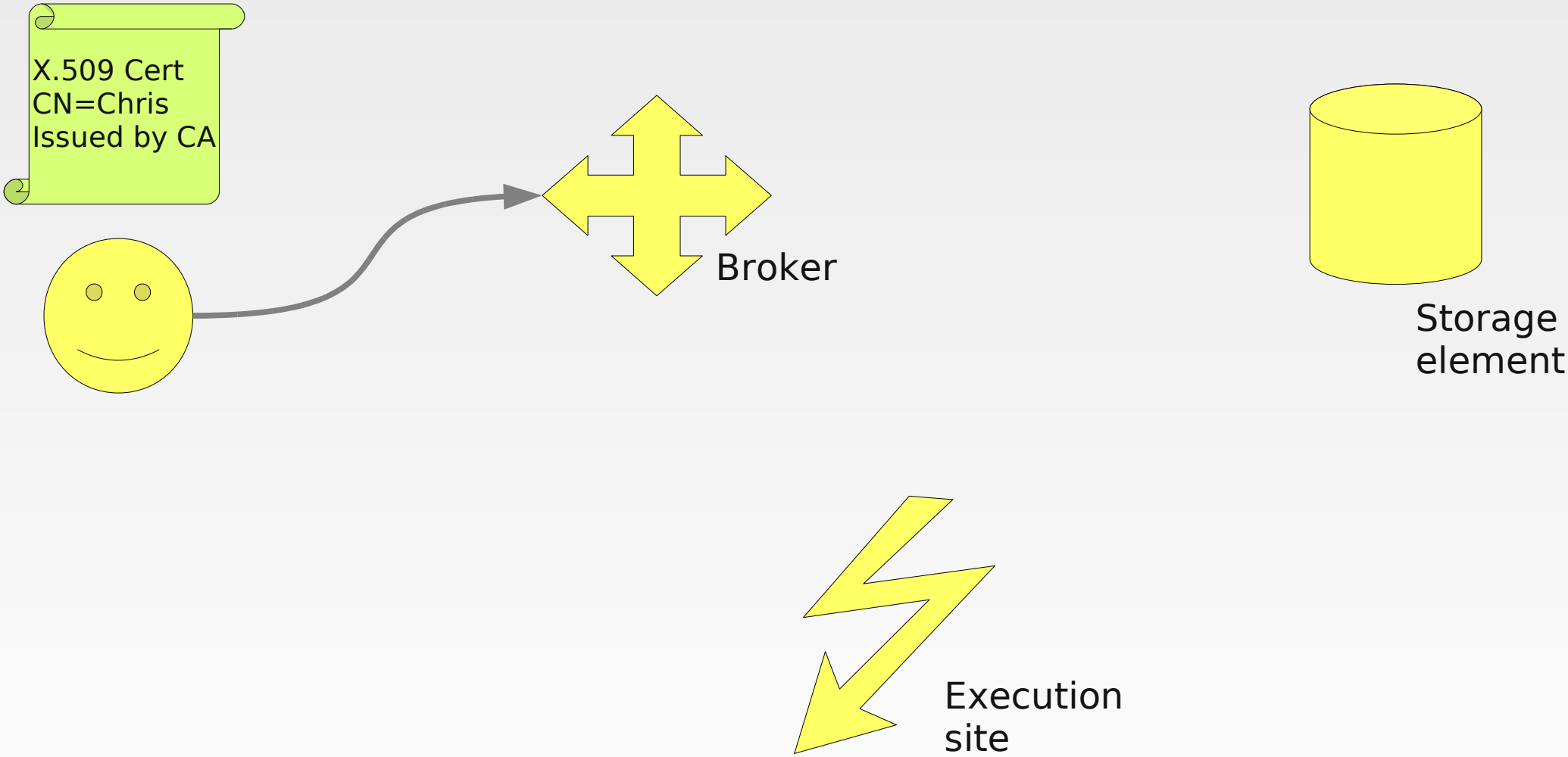
Proxy certificates - example

- Proxy certificates use impersonation: the service receives the credentials which resemble original user's credentials:

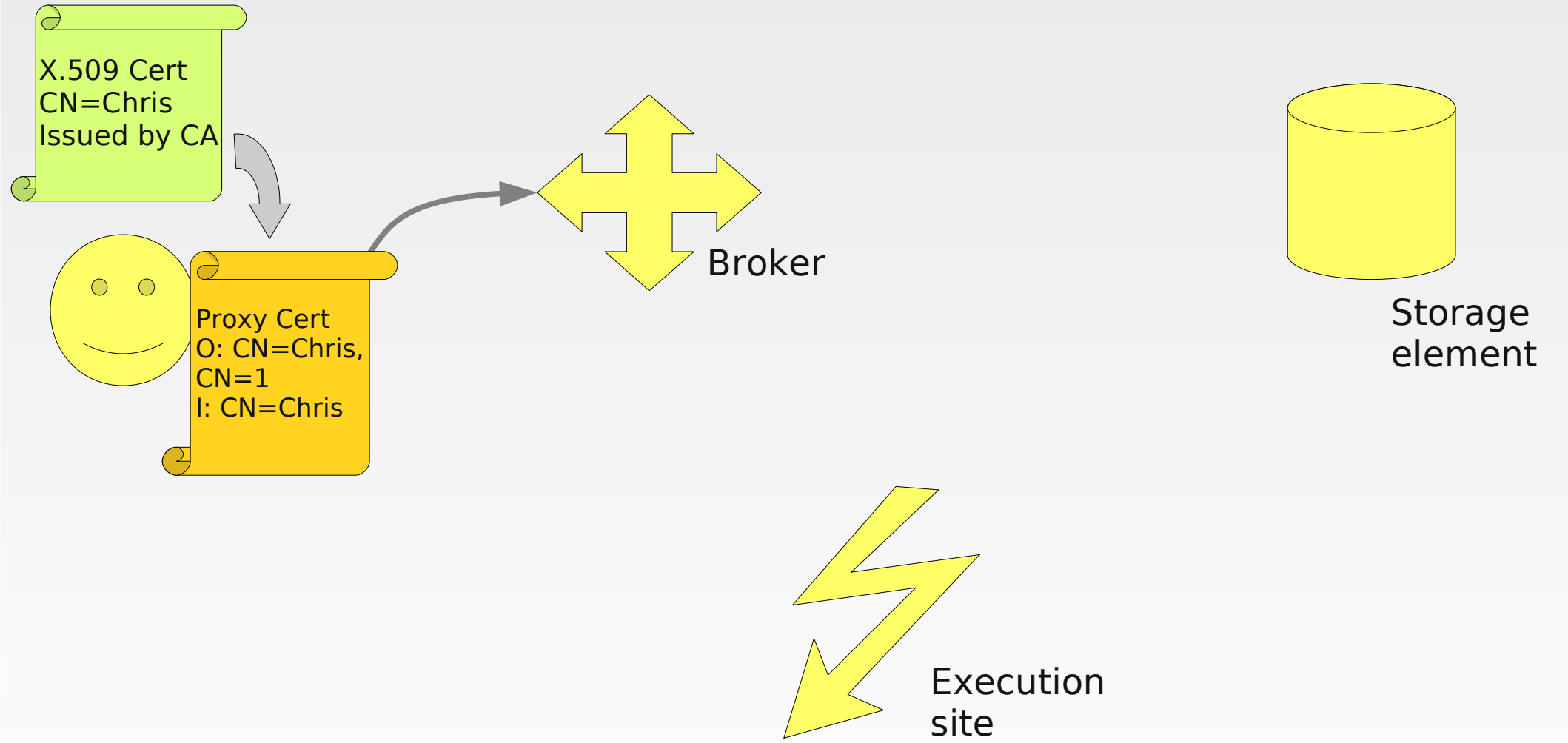
```
[golbi@i2ui ~]$ grid-proxy-init
Your identity: /C=PL/O=Grid/O=ICM/CN=Krzysztof Benedyczak
Enter GRID pass phrase for this identity:
Creating proxy ..................................... Done
Your proxy is valid until: Wed Apr 28 01:33:48 2010

[golbi@i2ui ~]$ grid-proxy-info
subject  : /C=PL/O=Grid/O=ICM/CN=Krzysztof Benedyczak/CN=213687476
issuer   : /C=PL/O=Grid/O=ICM/CN=Krzysztof Benedyczak
identity : /C=PL/O=Grid/O=ICM/CN=Krzysztof Benedyczak
```

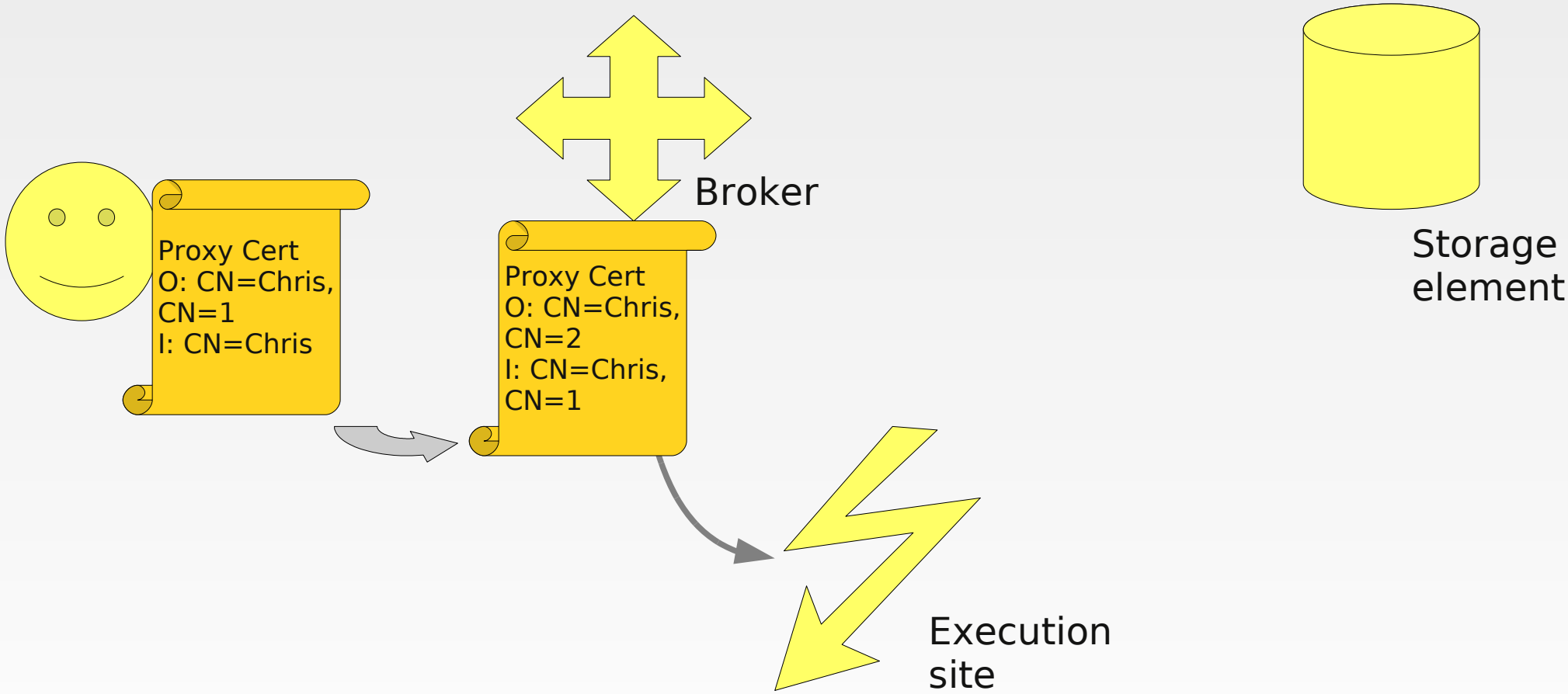
TD with proxies



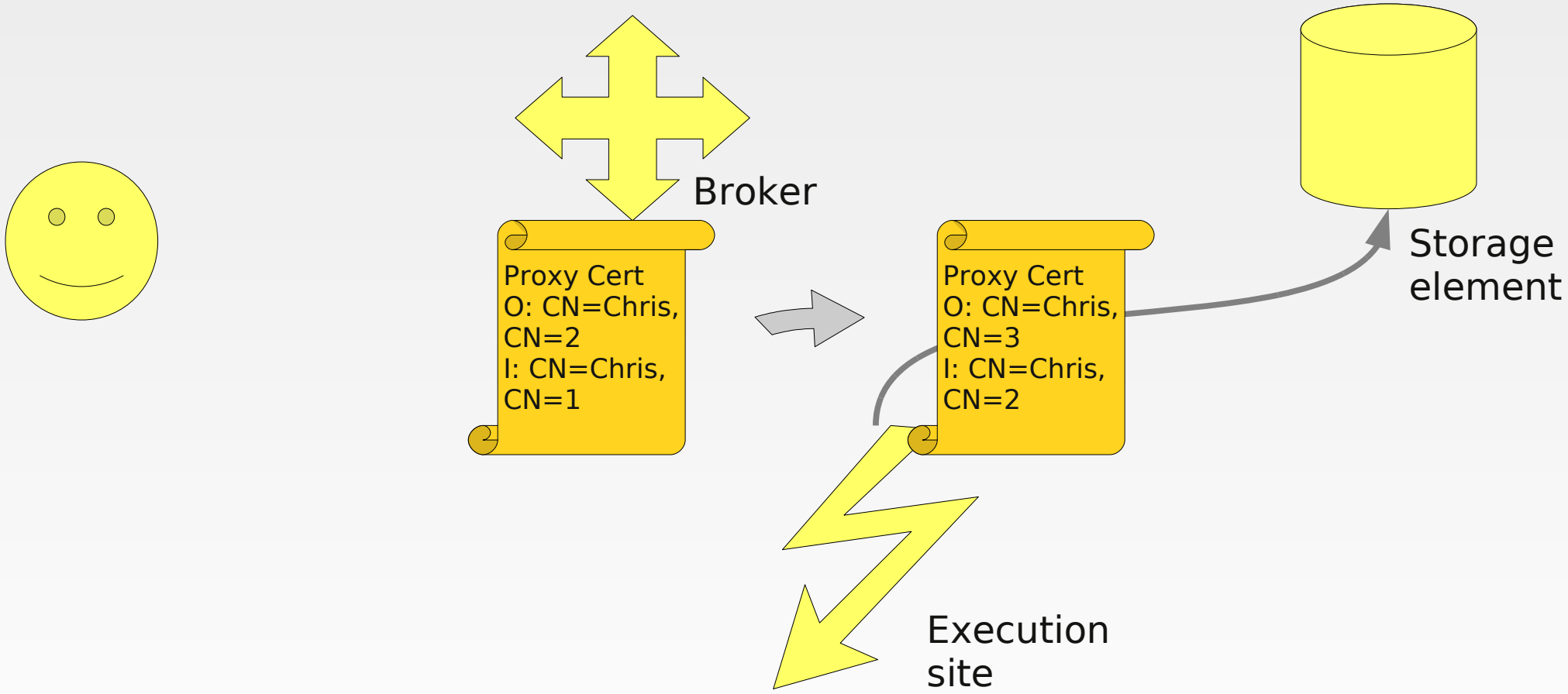
TD with proxies



TD with proxies



TD with proxies

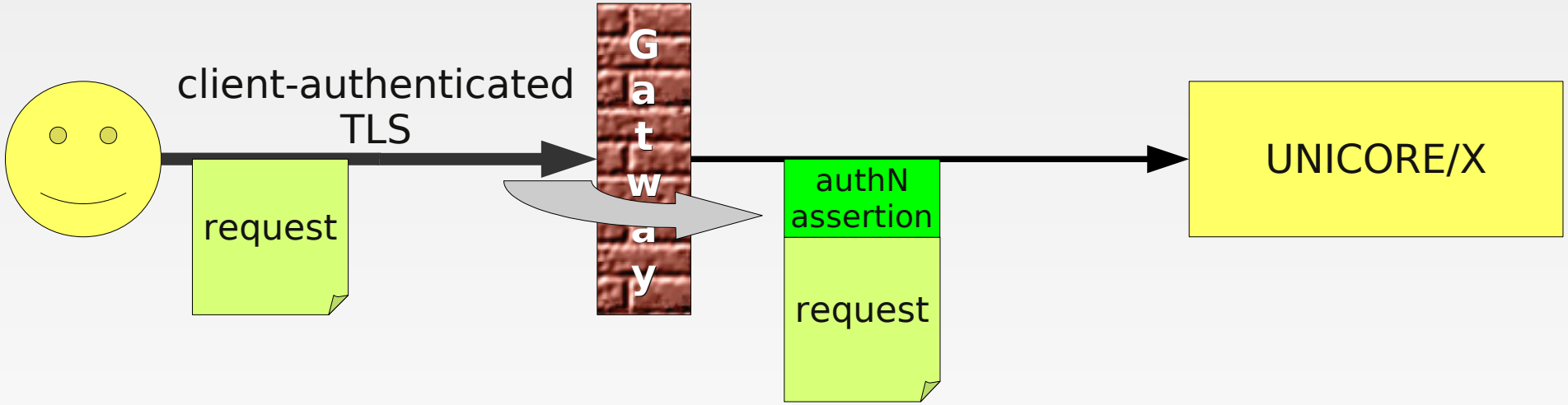


Proxy certificates: selected problems

- Standards support. There are three flavors of Proxy certificates: legacy, pre-RFC, RFC 3820.
- Private key protected by FS rights only: solved by short validity (24-48 hours).
- Short validity is a problem in case of long-running jobs. Solved by an additional infrastructure for renewing proxies.
- Complicated renewal: must be performed, different and incompatible tools used.
- Impersonation blurs the delegation chain. It is hard (if not impossible) to answer a trivial question: through which sites the request came?

UNICORE authentication

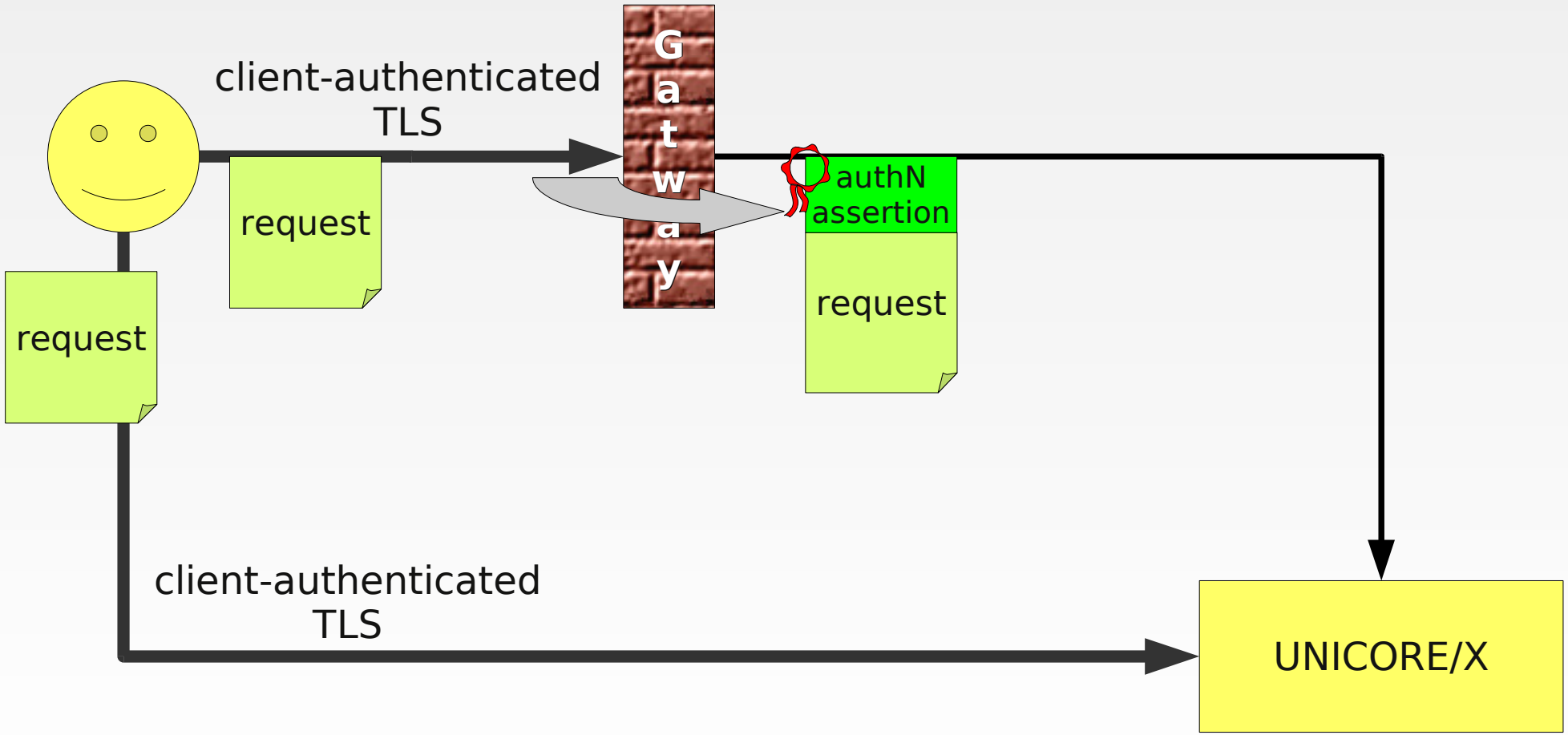
- In UNICORE authentication is strictly X.509 SSLv3/TLS based.
- Authentication can be performed by a Gateway or by a UNICORE/X site if direct access is permitted.



- Gateway functions as a single point of entry: only one firewall port need to be opened.

UNICORE authentication

- If direct access is permitted Unicore/X site can't trust that authentication assertion is genuine. It must be signed by gateway and signature must be checked by a site.



Explicit Trust Delegation

- Explicit Trust Delegation (ETD) was introduced in UNICORE 5.
  - The paper by D. Snelling, S. van den Berghe and V. Li: "Explicit Trust Delegation: Security for Dynamic Grids", available from: http://www.fujitsu.com/downloads/MAG/vol40-2/paper12.pdf provides a detailed description.
- UNICORE 5 ETD and UNICORE 6 ETD differ significantly!
  - UNICORE 5 ETD can be considered a static ETD while UNICORE 6 ETD is really dynamic.
- First of all the original ETD defined that three different entities may be bound to each grid job:
  - Consignor: the entity which actually sent a job.
  - Endorser: the entity which authorized the job.
  - User: the entity on whose behalf the job was submitted.

ETD in UNICORE 5 in action

- Endorser was the one who signed part of the job (a sub task).
- UUDB was used to explicitly state who can send jobs on somebody's behalf.
- As a result:
  - complicated to understand (difference between endorser and user, who should be authorized)
  - static as all trust relationships had to be manually entered into UUDB (OK for portal as a single point of entry to a grid but unsuitable for regular users).

ETD in UNICORE 6

- Endorser role is "discriminated".
  - It could be used in future however it seems we don't have important reasons for implementing this concept.
- User and consignor roles are the primary concepts.
- Consignor creates and sends a request.
  - It is a client in client-server model.
  - Server establish who is the request's consignor by means of authentication.
- User is the principal on whose behalf the request should be invoked.
  - I.e. the request should be invoked with all User's permissions.
  - Typically the request is related to a job which was initially initiated by the User.
  - By default User==Consignor.
  - It is similar to "effective user (id)" in UNIX systems.

User selection

- Each and every Consignor can request that operation it is invoking should be performed on behalf of an arbitrary User.
- Technically this is performed by adding a special token, called User assertion into a SOAP header.
- The User assertion is unsigned.
- Basically speaking it is a wish.

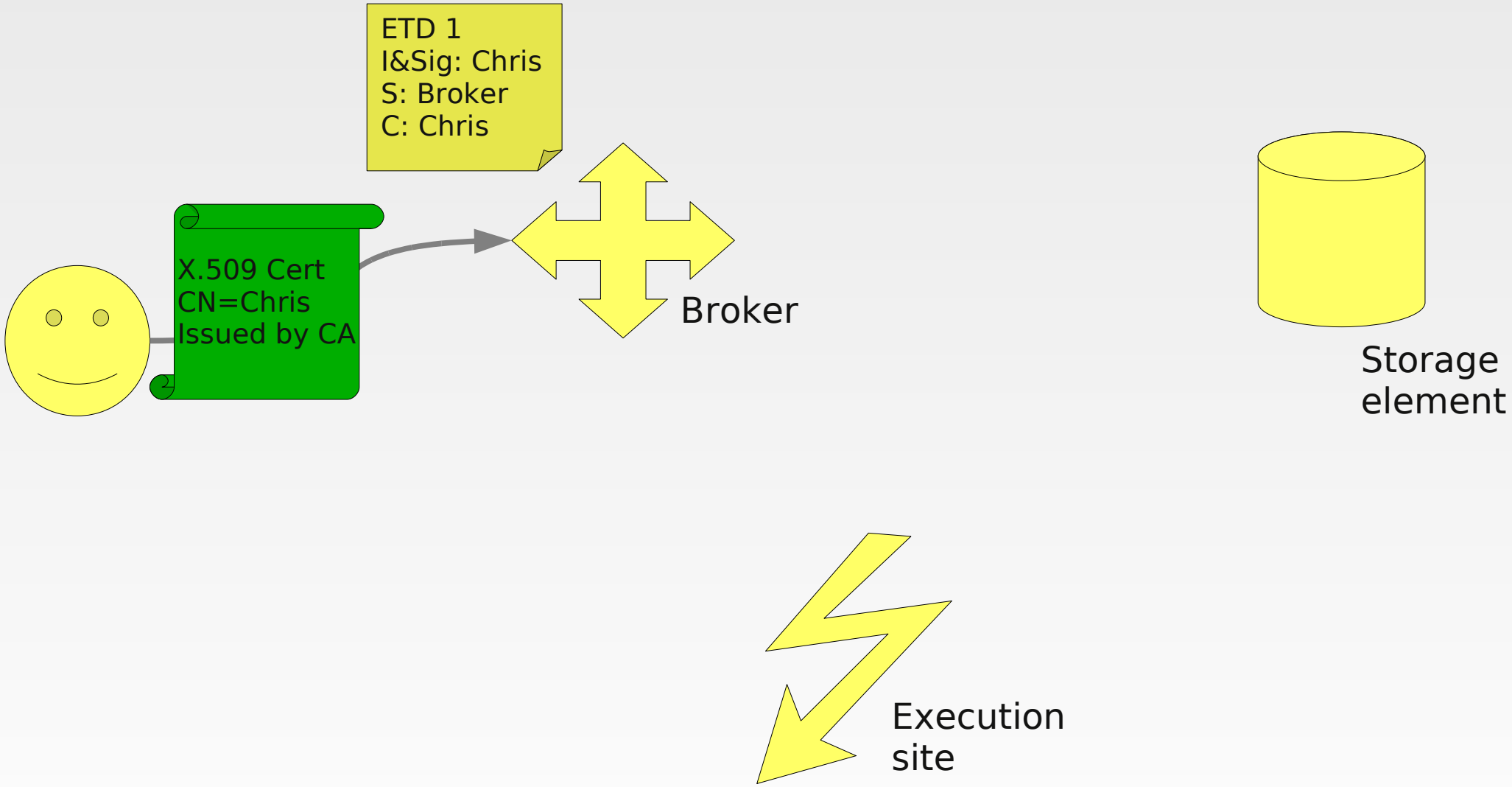
User selection approval

- Site which receives a request with a User assertion must somehow verify if a Consignor is allowed to perform operations on the User's behalf.
  - Consignor==User is always allowed.
  - If Consignor has a special role called 'trusted-agent' then user selection is accepted. This mimics the UNICORE 5 model.
    - Note that this feature is not widely used (if anywhere) and not well tested.
  - Otherwise the Consignor must present a **valid trust delegation assertion, issued by the User**.

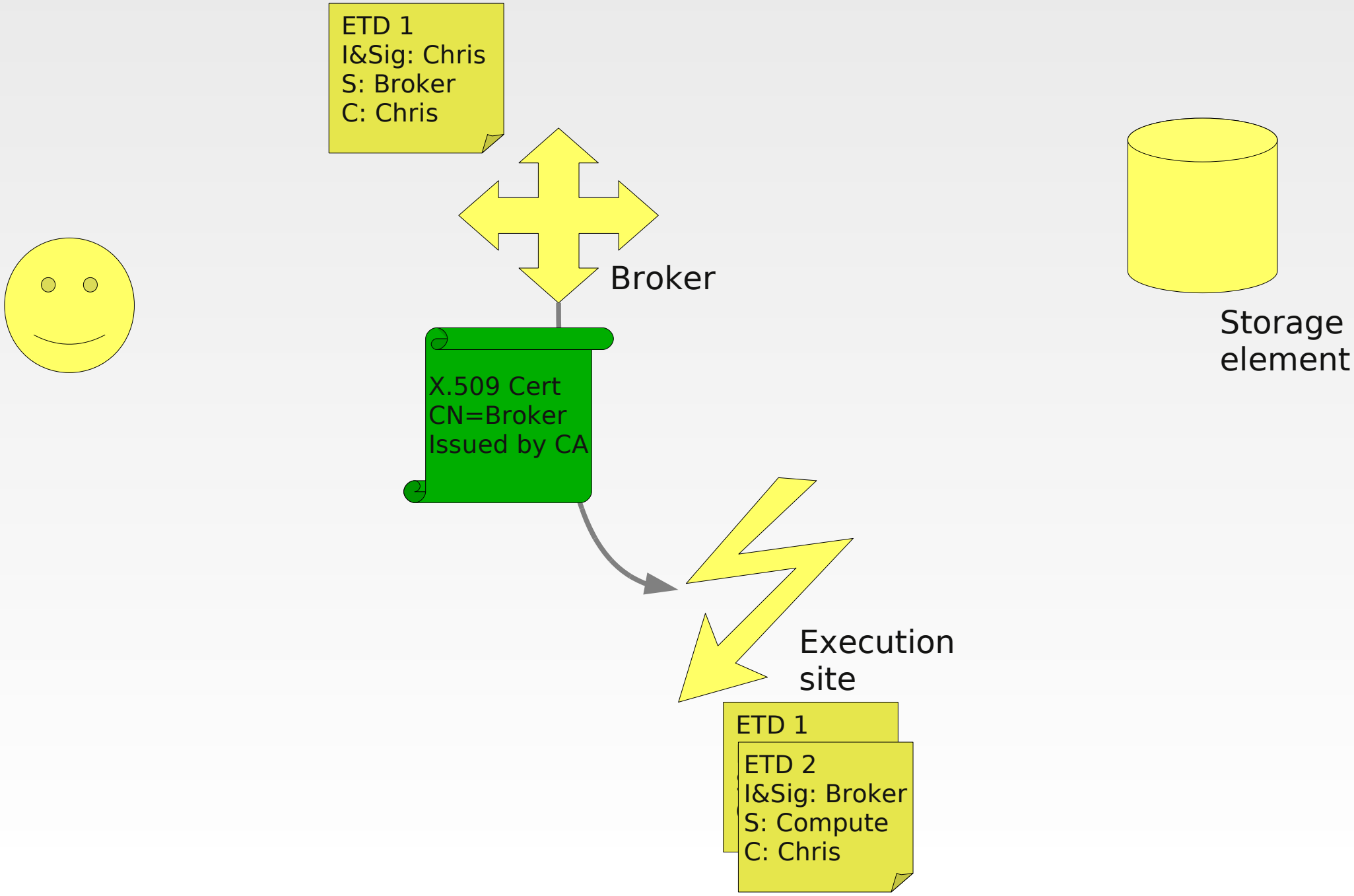
Details of ETD

- ETD assertion contains the following data:
  - the **trust delegation issuer** (who is the user in case of a single trust delegation),
  - the **delegation subject**,
  - the validity time frame of the assertion,
  - other usage restrictions,
  - a special token which confirms that the whole document is a trust delegation,
  - an initial trust delegation issuer called a **trust delegation custodian** (i.e. the user),
  - signature made by the issuer.

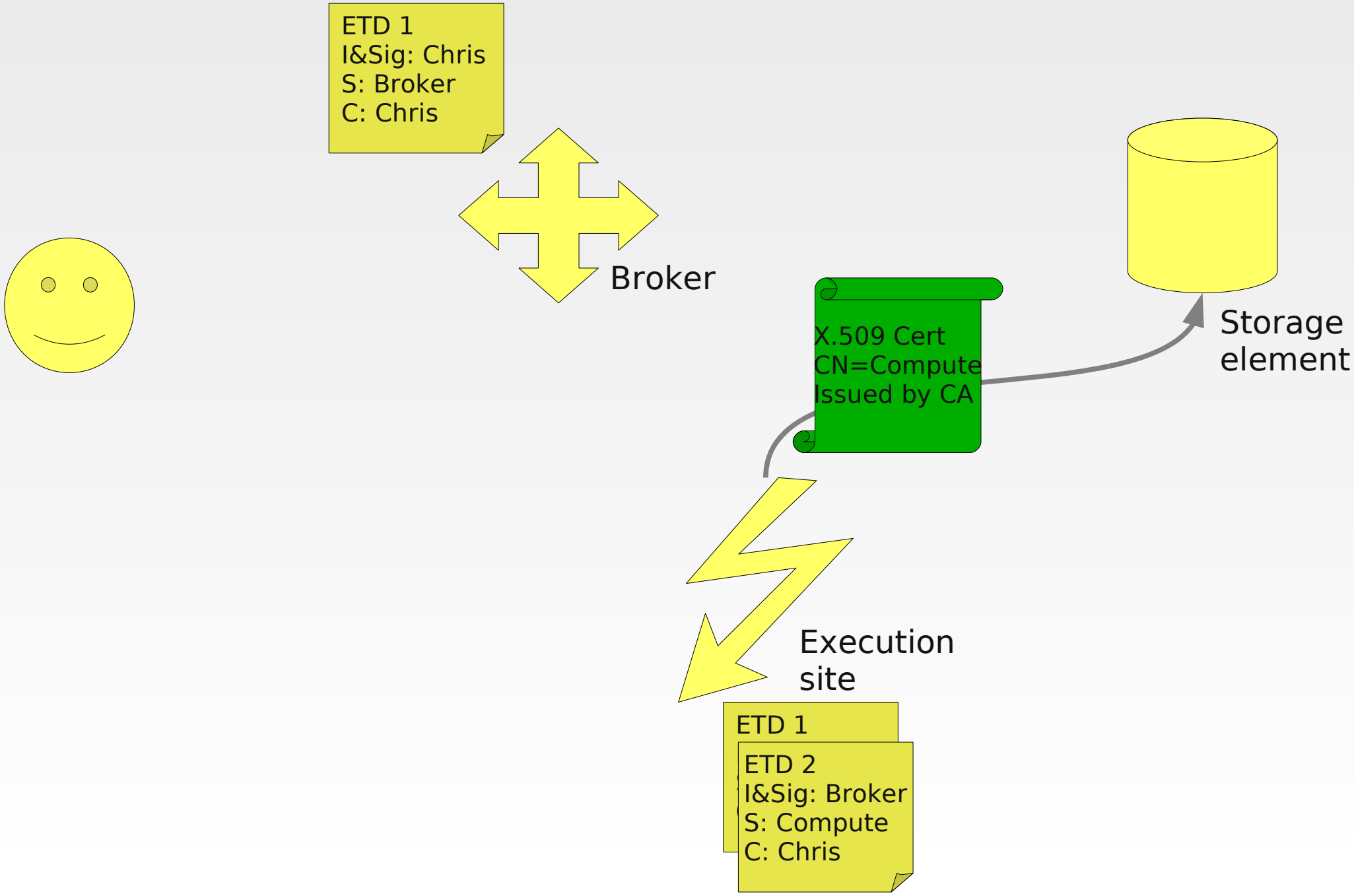
TD with ETD


ETD 1
I&Sig: Chris
S: Broker
C: Chris
X.509 Cert
CN=Chris
Issued by CA
Broker
Storage
element
Execution
site

TD with ETD


ETD 1
I&Sig: Chris
S: Broker
C: Chris
Broker
X.509 Cert
CN=Broker
Issued by CA
Storage
element
Execution
site
ETD 1
ETD 2
I&Sig: Broker
S: Compute
C: Chris

TD with ETD


ETD 1
I&Sig: Chris
S: Broker
C: Chris
Broker
X.509 Cert
CN=Compute
Issued by CA
Storage
element
Execution
site
ETD 1
ETD 2
I&Sig: Broker
S: Compute
C: Chris

ETD verification

- To verify a single ETD assertion quite simple algorithm is used: signature must be valid, and must be made by the issuer.
  - Also all restrictions as validity time should be respected.
- As we could see the ETD assertions may be chained. To verify the chain the following rules are used:
  - All chain assertions must have a common custodian equal to the expected custodian.
  - An initial assertion issuer must be equal to the chain custodian.
  - For every assertion, except the initial, with issuer A, there is an assertion with the A subject (generally it is the previous assertion in the chain).
    - It means that as the delegation is passed along the chain, the subject of a delegation assertion becomes the issuer of the next assertion in the chain.
- Without a custodian field a malicious site could produce fake "trust delegation" chain by combining two unrelated TDs.

Digital signature and non-repudiation

- UNICORE security stack guarantees job's non-repudiation.
  - Non-repudiation ensures that job submitter can not deny that it actually submitted the job.
- The non-repudiation is achieved by requirement of a digital signature for key operations.
- As digital signature checking is an expensive operation it can be disabled.
- Digital signature is always done by a consignor.
- Currently the following actions require a digital signature:
  - TSF: CREATE TSS
  - TSS: SUBMIT job
  - SMS: DELETE, RECEIVE, RENAME, SEND, IMPORT, EXPORT,
  - WSRF: DESTROY, SCHEDULE_DESTROY.

Technical realization

- Client must know the DN of the ETD receiver. Obvious but...
  - not trivial as a client typically can talk only to the gateway while the trust must be delegated to a service. UNICORE publishes DNs of services in EPRs stored in registry.
- User, Gateway-AuthN and ETD assertions are all encoded as SAML attribute assertions with a predefined attribute used to encode an additional data and to distinguish them.
  - User assertions can carry additional consignor's preferences regarding a request.
  - Gateway-inserted authN assertion is always the first assertion due to several implementation reasons.
- This approach has two drawbacks:
  - It must be guaranteed that "attributes" from the special assertions are not mixed with a normal SAML attributes which may be pushed by a client for authorization.
  - It would be more logical to use SAML Authentication assertion for the Gateway authentication statement.

Pros and cons of the UNICORE approach

- Advantages of the presented model (in comparison to Proxy certs):
  - ETD assertions do not carry a sensitive data and therefore can have a longer validity then Proxy. No need to develop extension strategies.
  - The system is transparent – it is clear who does what and on whose behalf.
- Disadvantages (in general):
  - Currently nearly all WS operations require an additional data: User assertion and ETD assertion(s). This is a significant processing overhead.
  - X.509 infrastructure is very complicated and end users do not understand them. With ETD it is even harder.
  - Even if somebody do understand the PKI, it is quite cumbersome to use multiple computers and to renew (usually each year) the X.509 certificate.

Outlook for the future

- Eliminate the need to send the ETD/User assertion each time.
  - May be achieved by creating a security session (or association).
  - Require universal tools to support server and client side (especially caching of ETD assertions).
- Introduce Sort-Lived-Certificates. SLCs can resolve some of the general X.509 flaws.

Thank you!
Questions?

This work was supported by Kardionet project.