Mellon certificate authority

Our certificate authority (CA) is implemented with our Mellon and associated services. The Mellon is a crypto USB stick which has our private CA certificates stored. Like this the probability of leaking the private certificates is reduced by a lot. There is no API to read the certificates back from the Mellon so they can only be "read" using security holes in the underlying firmware library or through side channels.

There are three programs that come with the Mellon:

1. mellon-cli: A command line tool to access the Mellon locally.
2. mellon-api: A service providing a REST API for the Mellon.
3. mellonbot: A command line tool to communicate with the mellon-api service.

Using mellonbot or mellon-cli, certificates can be signed. We discuss how this can be done using mellonbot here. mellon-cli should be self-explanatory. Just execute `mellon-cli --help` for a description of the available commands.

Generate and sign a certificate

Certificates can be signed using `mellonbot`. `mellonbot` can be installed through our APT repository and is available precompiled for Debian and Ubuntu.

Generate and sign a certificate using mellonbot only

To generate and sign a certificate using mellonbot only, execute:

```
mellonbot -c /etc/mellon/your-cert.crt -k /etc/mellon/your-key.key -s 4 -m <CN>
```

Where `<CN>` can either be a string or JSON object. If a JSON object is passed, it is automatically checked for validity and base64-encoded for you. So for example:

```
mellonbot -c /etc/mellon/your-cert.crt -k /etc/mellon/your-key.key -s 4 -m 'it-works'
```

or

```
mellonbot -c /etc/mellon/your-cert.crt -k /etc/mellon/your-key.key -s 4 -m '{"test":"It works!"}'
```

Note that the content of the JSON is not checked!

Sign an externally generated certificate

To generate a certificate, execute for example:

```
certtool --generate-privkey --ecdsa --curve secp521r1 --outfile privkey.pem
```

To create a certificate signing request, execute the following command and fill in the common name (if it is a JSON object, you need to base64-encode it before passing it here!). Do not fill any other fields.

```
certtool --generate-request --no-crq-extensions --load-privkey privkey.pem --outfile request.pem
```

And finally to get the signed certificate, execute:

```
mellonbot -c /etc/mellon/your-cert.crt -k /etc/mellon/your-key.key -s 4 -l request.pem
```

Create a user client certificate with permissions to sign certificates

For a user to be able to sign certificates, an administrative certificate is required. This certificate must be signed by our root CA number 5. There are no requirements on the provided certificate information. You can enter anything here.

After the certificate has been signed, it must be imported into the mellon-api service, by using the following command on the server:

```
mellon-api -i certificate.pem -a '<ACL>'
```

Please note that the mellon-api service must be stopped to do this.

The ACL specifies what this certificate is allowed to do. It must be passed as a JSON. There a three properties:

Property name	Description
`pathAccess`	A JSON object with REST paths the certificate is allowed to access. The property name is the REST path, the property value is a `String` `Array` with the allowed HTTP methods. For example: `{"/api/v1/mellon": ["GET"], "/api/v1/ca": ["GET", "POST"]}`. You can use `*` as a wildcard for the REST path.
`typeSignAccess`	An `Array` of certificate types this certificate is allowed to sign. E. g. `apartment`, `c1-core`, `module`, etc. The type is set in the `type` property of the

Property name	Description
	CN JSON (see section about client certificates). You can use `["*"]` as property value to allow signing of all certificate types.
tokenCreationAccess	An `Array` of certificate types this certificate is allowed to create tokens for. A token can be exchanged into a certificate by a third party without having an administrative client certificate. You can use `["*"]` as property value to allow token creation for all certificate types.

Example JSON

The following JSON specifies a full access ACL.

```
{
   "pathAccess": {
      "*": ["GET", "POST"]
   },
   "tokenCreationAccess": ["*"],
   "typeSignAccess":["*"]
}
```

An example import command using this JSON might look like:

```
mellon-api -i certificate.pem -a '{"pathAccess":{"*":["GET","POST"]},"tokenCreationAccess":
["*"],"typeSignAccess":["*"]}'
```

The private key must be stored GPG-encrypted using a hardware security module like the Yubikey. To encrypt the certificate file, execute:

```
gpg -e -a private-key.pem
```

After the private key has been encrypted, delete the unencrypted file.