(Sub)CA contract model, pseudocode

TYPES:

Certificate:

* Contains all information necessary to recreate standard certificate cer file.
* Contains extension with address of issuing CA, or 0 if self-signed
* Contains asymmetric signature.
* It’s validity is assured by any of the following
  + thru certificate signature (mandatory)
  + thru presence of the hash found in the address of the issuing CA
* No other changes are necessary

**Data:**

Address owner; //may be derived from actual public key, but it is not necessary, populated

//on creation

Certificate thisCA;

Dictionary<hash, certificate> publishedCertificates; //Certificates, that are intentionally public

Hash[] signedCertificates; //hashes of all certificates issued by this CA

Hash[] revokedCertificates;

SignService signService; //populated on creation, not sure how this works internally

**Public Methods (owner only):**

void Blacklist(Hash hash):

revokedCertificates.Add(hash)

cert IssueEndCertificate(CertificateSignatureRequest csr, bool publish):

//offline CSR check as appropriate. Note that csr is also delivered offline

Obj o=Transform(csr, this.address)

Cert cert=SignService.Sign(o) //send to offline signing service

signedCertificates.Add(HashCert(cert))

if (published)

publishedCertificates.Add(HashCert(cert), cert)

return cert;

cert IssueCA(CertificateSignatureRequest csr, Address csr, bool publish):

//offline CSR check as appropriate. Note that csr is also delivered offline

Obj o=transformCA(CSR, this.address, thisCA)

contractAddress=CreateContractForCA(address, o) //make this contract for given cert

o=AddAddressToCA(o, contractAddress)

Cert cert=SignService.Sign(o) //send to offline signing service for crypto signature

signedCertificates.Add(HashCert(cert)) //blockchain signature

if (published)

publishedCertificates.Add(HashCert(cert), cert)

return cert

(methods for signing documents, encryption etc, using the oraclized crypto?)

Public Methods(anyone):

//note, that I assume, these can be executed for free locally. These functions are basically optional.

Cert GetCert()

return thisCA

Bool Verify(cert, Address[] trustedCAs)

//Checks cert validity

//climbs thru the chain and searches for trusted CAs to validate trust.

//Certificate signature not verified and whitelist is used instead (signedCertificates)

Private methods:

//just descriptions

TransformAdds our extension with CA contract address to the certificate, removes the selfsignature etc., prepares for actual signing.

TransformCAs above, but adds also reference to self in the blockchain to the extension.

CreateContractForCA. Creates this contract, populates owner(!), uploads the signed cert, once it is signed (not in pseudocode above)

Note on crypto service. I presume, this can be configured to use the private key for the certificate securely.