Post Quantum Migration

How to migrate X.509 / TLS from one Public-Key Algorithm to a new one

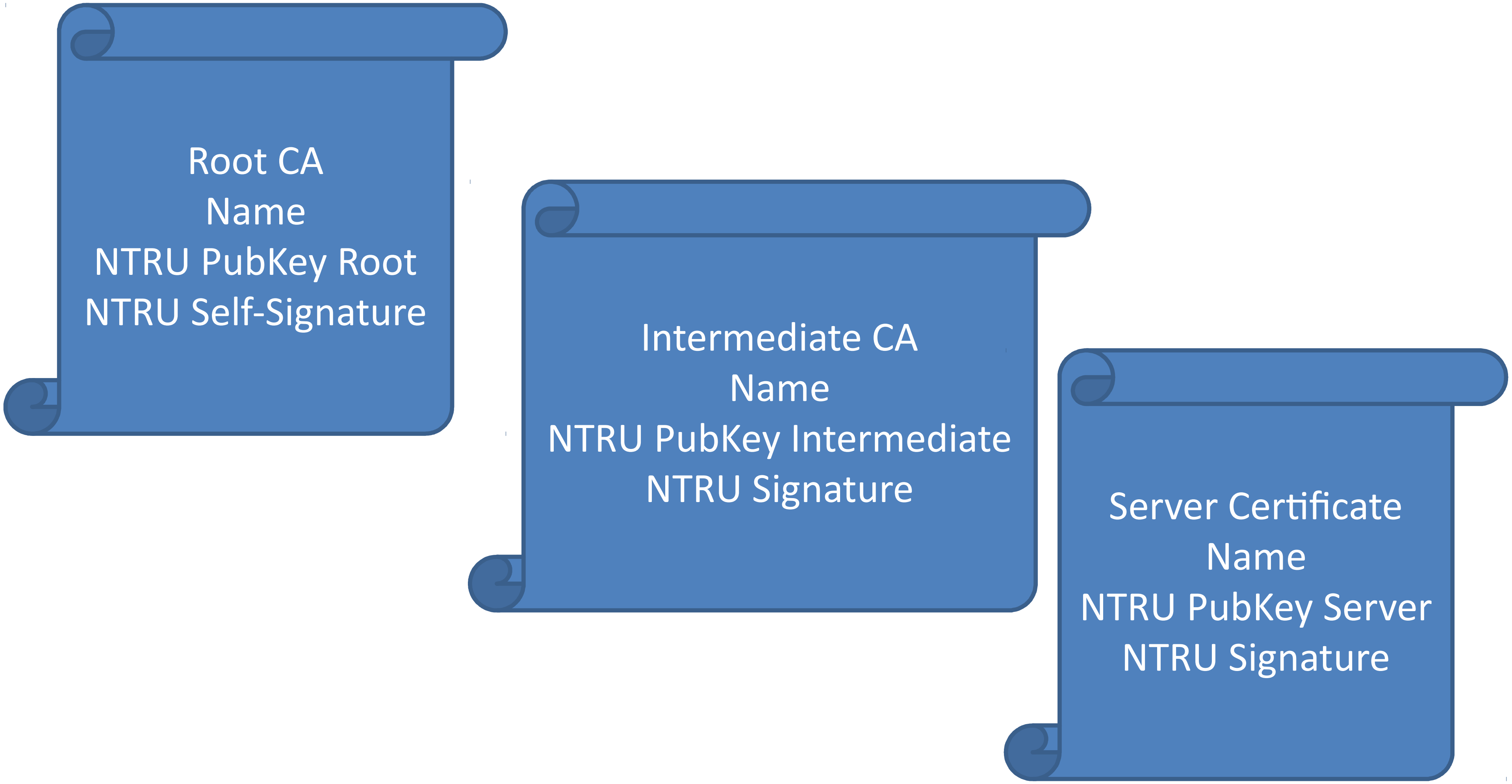
Current Situation with RSA

Root CA
Name
RSA PubKey Root
RSA Self-Signature

Intermediate CA
Name
RSA PubKey Intermediate
RSA Signature

Server Certificate
Name
RSA PubKey Server
RSA Signature

What if we just use NTRU instead?

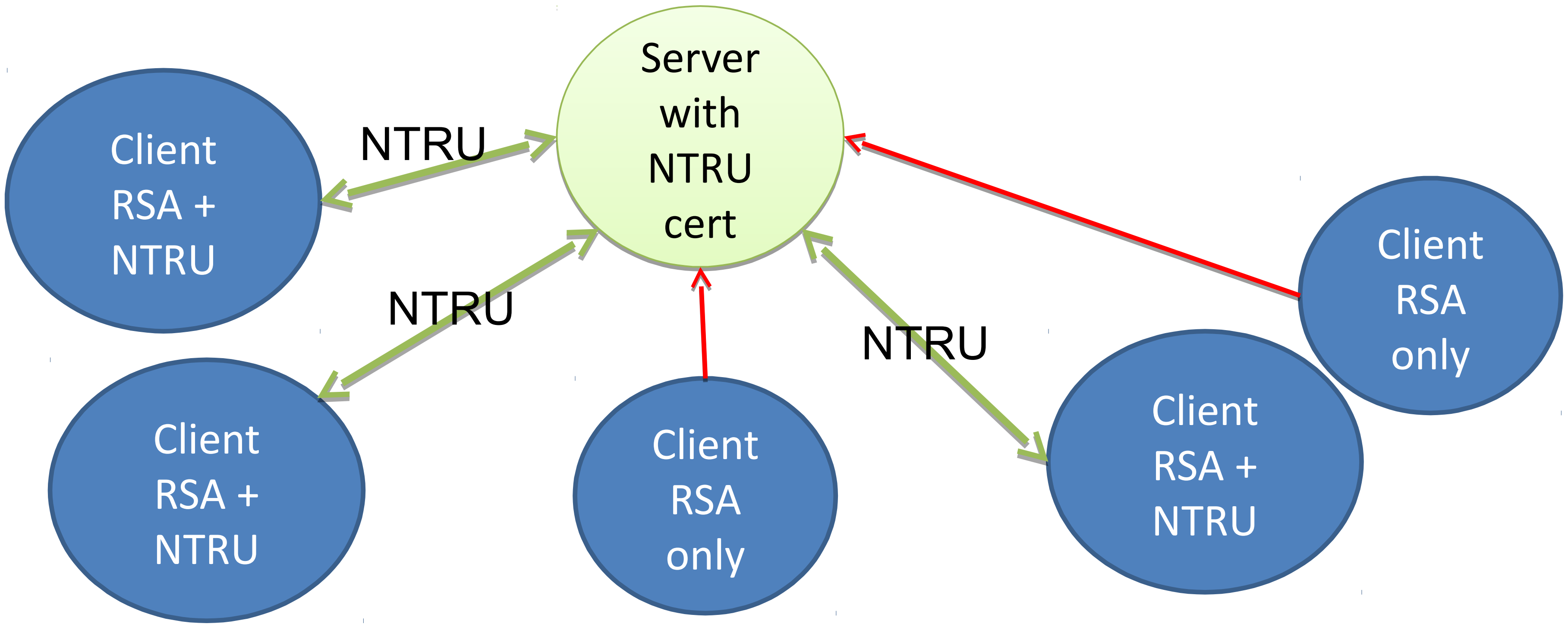


What if we just use NTRU instead?

- Every CA would create new NTRU root certificates and start issueing new certificates from the new root
- Servers would deploy new certificates with the new NTRU root
- Clients would get new root certificate lists

Server Interoperability

- When a server only has a NTRU cert
- it cannot communicate with old RSA clients



What if we just use NTRU instead?

- If the server would install a new NTRU certificate, but the client would still not be NTRU capable and have the NTRU root certificate installed, the client could not communicate with the server anymore.
- Therefore the server operator will replace a RSA certificate with a NTRU certificate only when 99.9% of the clients will support NTRU, otherwise they will have huge customer-losses and support efforts.

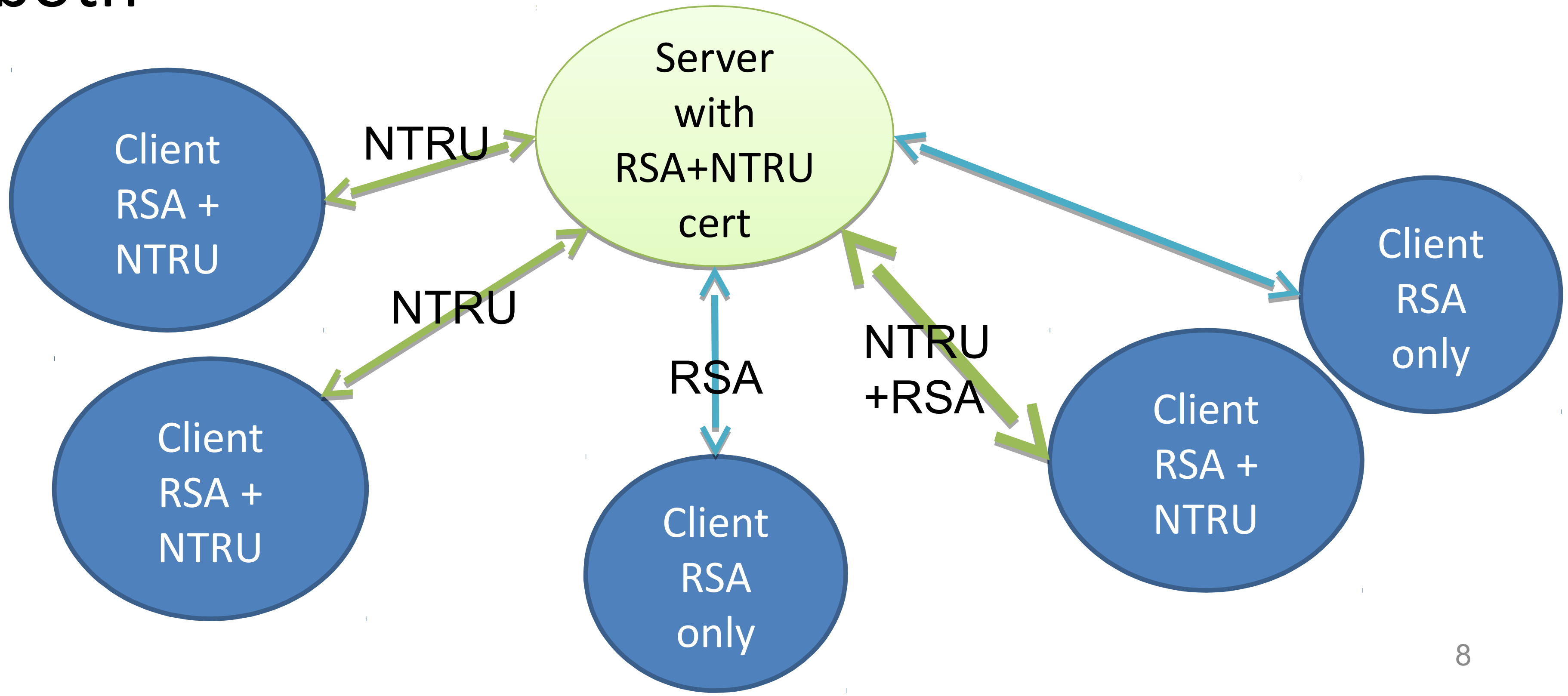
What if we just use NTRU instead?

When will 99,9% of the clients support NTRU?

- On Windows with the next operating system upgrade
  - Operating system upgrades are usually done when the hardware dies (see Windows XP)
    - At least 10 years on the public internet
- On Linux and some other desktop operating systems
  - perhaps a bit earlier

Our goal

- When a server has both RSA and NTRU
- old clients can use RSA, new clients can use both



Variant A:

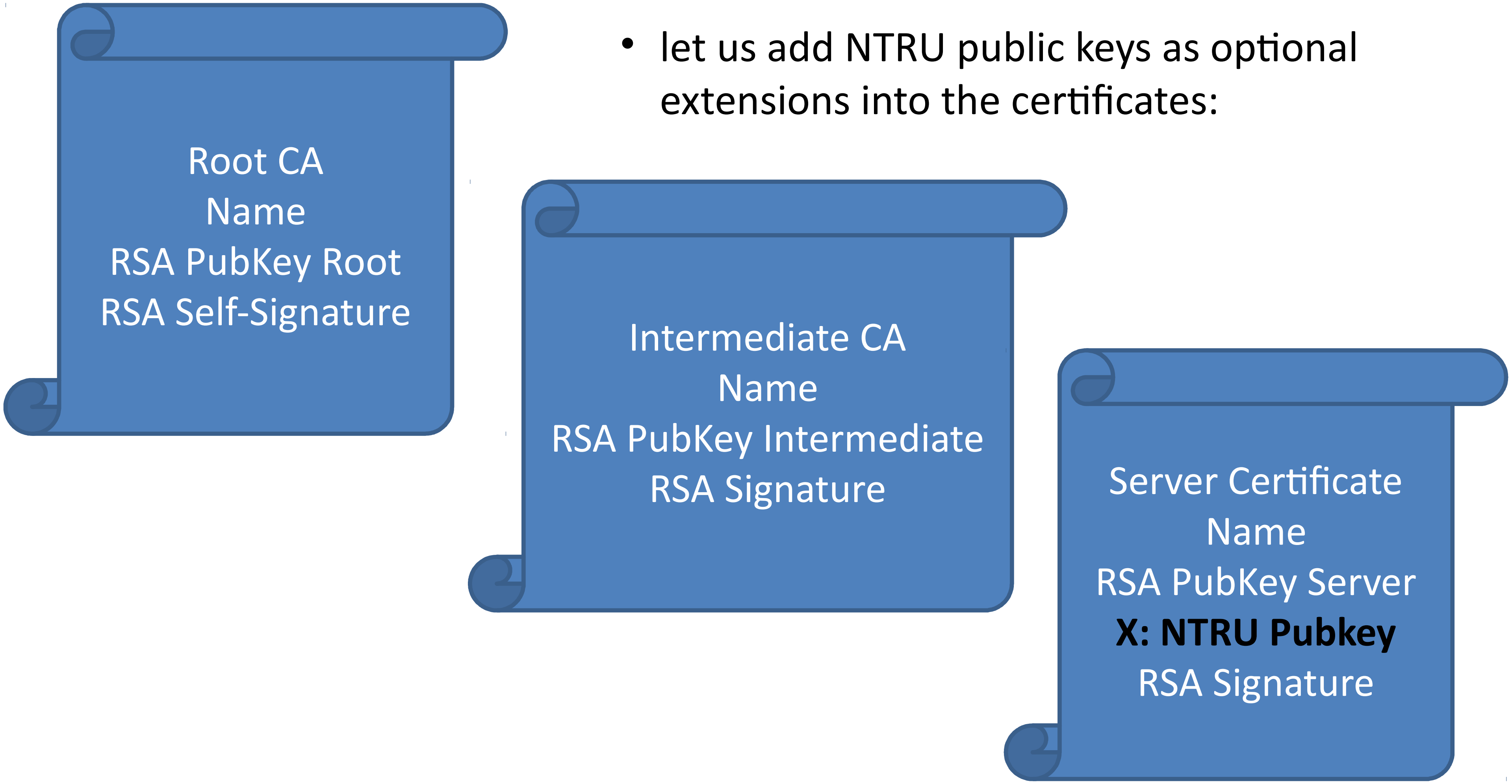
- We could have the CA´s issue both NTRU and RSA certs
- and have the users acquire both certificates and deploy both certificates into their webservers
- then the webserver could automatically select the right certificate depending on the ciphersuite advertised by the client.

Drawbacks Variant A:

- complex to configure for the user
- confusing for the users (and PKI is already too confusing)
- Double costs for 2 certificates
- none of the servers support ciphersuite-dependent certificate selection yet
- And what about S/Mime? How do you transport both certificates to the peer?

Variant B: Additional Public Keys

- let us add NTRU public keys as optional extensions into the certificates:



Additional Public Keys

- are included in a certificate
  - as a non-critical (which means that it´s optional and can be ignored if it is not understood) extension
- Old clients that do not understand NTRU yet, or do not have the NTRU root certs, can continue communicating with RSA.
- Newer clients that support NTRU can start using them

Additional Public Keys

- are included in a normal RSA certificate
- therefore there are no additional costs for secondary certificates.

- Specification:
- http://www2.futureware.at/PostQuantum/AdditionalKeyDraft.txt
- Demo: http://www2.futureware.at/PostQuantum/AdditionalPublicKey/

Additional Public Keys

- they can be integrated in an automatic way:
  - certificate request generation software can automatically generate both RSA and NTRU private keys and generate certificate signing requests with both keys in it, certificate authorities can automatically issue certificates for both keys, and software that knows about it can use both to provide assurance against Shor attacks.
- There is no additional user-interface necessary

Disadvantages

- More complexity in the SSL/TLS/S-Mime Stacks to handle Additional Public Keys
  - But since we use exactly the same structure inside Additional public keys as normal Public Keys, it should not be too bad

Securing the whole CA chain:

Root CA
Name
RSA PubKey Root
optional Extension:
NTRU Pubkey
RSA Self-Signature

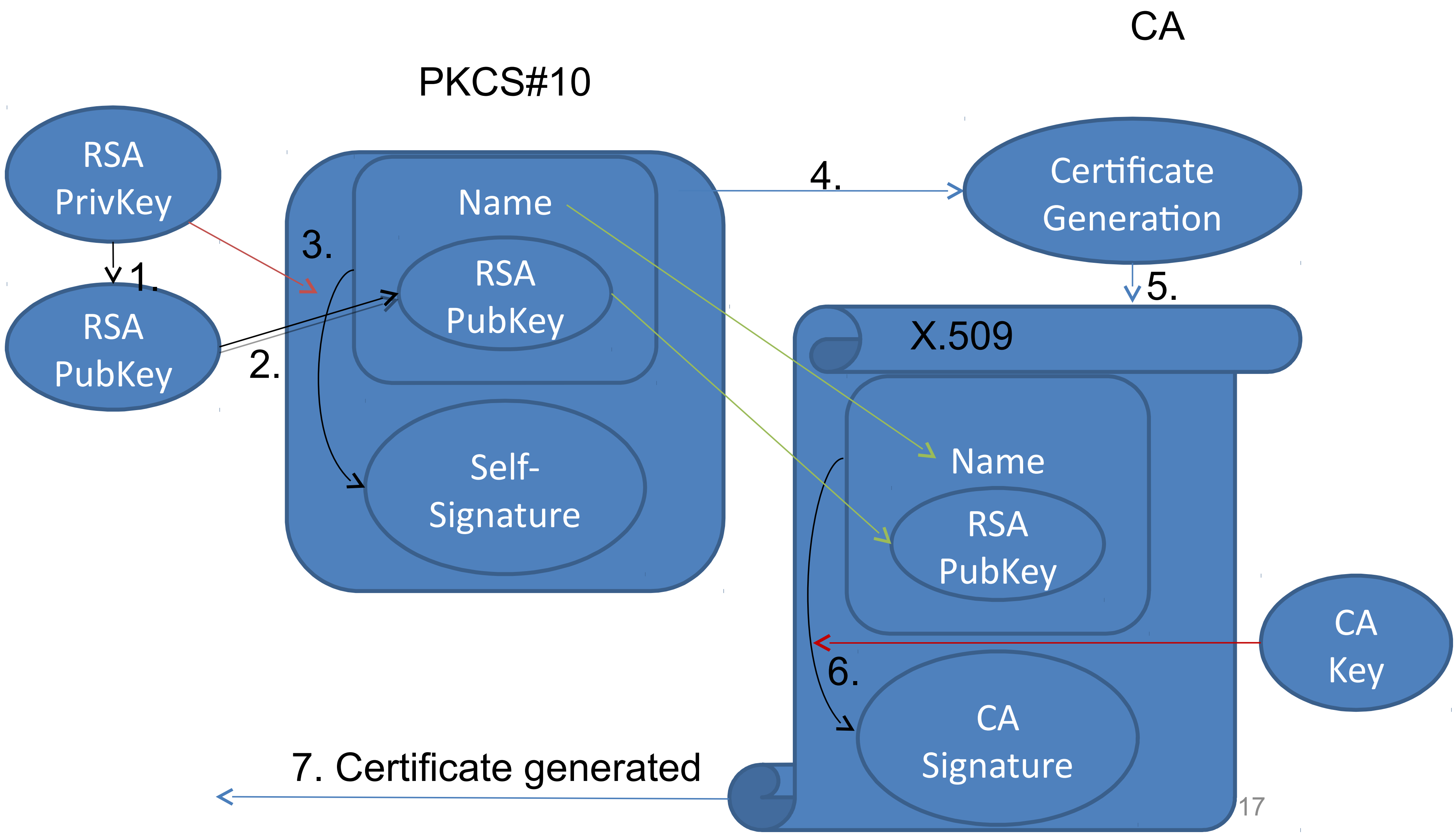
- It would be good if we could secure the whole CA chain, but unfortunately it is not very easy:

Intermediate CA
Name
RSA PubKey Intermediate
X: NTRU Pubkey
X: NTRU Signature
RSA Signature

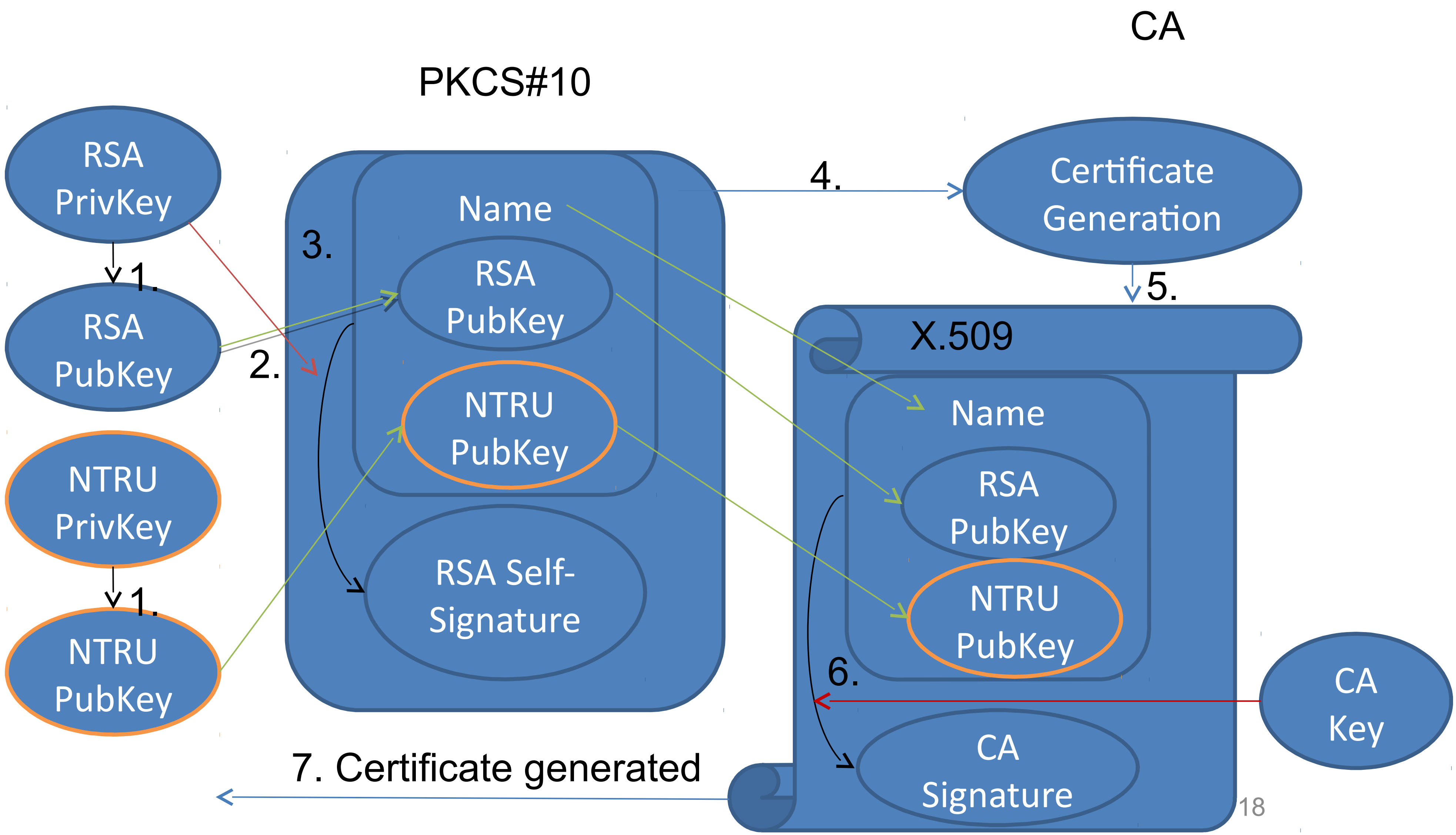
Server Certificate
Name
RSA PubKey Server
X: NTRU Pubkey
X:NTRU Signature
RSA Signature

Since the new signature is now part of the certificate that should be signed, it changes the document when we put it in, so it is difficult to define what exactly the signature has to sign.

Certificate Signing Requests (CSR)



Certificate Signing Requests (CSR)



Certificate Signing Requests (CSR)

- Integration of Additional Public Keys into PKCS#10 for CSR´s (Certificate Signing Requests):
  - As soon as the client supports it, it can generate CSR´s with NTRU public keys in it.
  - If the CA does not support it, it just issues a normal RSA certificate without NTRU, so we have a failsafe fallback, and clients can issue CSR´s with APK-NTRU opportunistically

Certificate Signing Requests (CSR)

- Unfortunately, we cannot have CSR´s self-signed with RSA+NTRU easily.
  - since it needs to be downward compatible so that CA´s that do not recognize Additional-Public-Keys can still use it as a normal CA and simply ignore/drop the Additional Public Key
- We could integrate an additional self-signature extension Additional-Self-Signature
  - what exactly would it sign?
  - It seems to be quite some effort for less effect, but we are open for proposals.

Certificate Signing Requests (CSR)

- Integration of Additional Public Keys into the CA software:
  - As soon as the CA supports it, the certificate contains NTRU
- Integration of Additional Public Keys into the SSL/TLS stacks in Browsers, Webservers, … as soon as the other client supports it, it can be effectively used.

Deployment proposal

- 2014/2015
  - Implement NTRU + Additional Public Keys capability into SSL stacks, Crypto Software, Browsers
    - Beware of potential incompatibel bitstream changes during 2014, until the bitstream is really frozen. (hopefully end of 2014)

NTRU current status

- NTRU-Encrypt
  - C Library API most likely stable
    - so you can start coding against the API
    - Java API currently being audited
  - Format for Public Keys: WILL CHANGE SOON
    - so please wait for it before you put NTRU keys into your root certificate
- PASS-Sign
  - Library API not complete yet
  - Bitstream not existing yet

Deployment: CA´s

- CA´s SHOULD start issueing NTRU certificates for NTRU CSR´s
- CA´s SHOULD start issueing additional public keys in certificates for certificate signing requests that include additional public keys

Deployment proposal

- 2014/2015 Browsers (and other CSR generating software) …
  - SHOULD start creating RSA + NTRU keypairs and generate RSA PKCS#10 CSR´s with NTRU as additional public key
  - SHOULD NOT start issueing NTRU-only CSR´s, unless the user really really wants it.

Deployment proposal

- At first NTRU can be supported optional.
- If and when Shor attacks become reality, the rules can be changed and NTRU can become mandatory.
  - Start with higher value targets, e.g. define a deadline for when EV certificates must contain a NTRU additional public key, to still be displayed "green"

Deployment proposal

- I guess that we can do a silent rollout within something like 3 years, augmenting RSA.
- Around 2017, we can start trying to issue NTRU-only root CA´s, and see how they perform.