Homework #8 Solutions

Brian A. LaMacchia
bal@cs.washington.edu
bal@microsoft.com

Question 1 – Fun w/ Revocation

- VeriSign's RSASecureServer.crl. As of 3am Wed., Feb. 22:
  - Valid from 2/22/06 to 3/8/06
  - 515,243 bytes in size
  - 14,714 entries
- Assume that all of the certs listed on the CRL were issued within the past 12 months.
- VeriSign claims to have about 500,000 sites with "Secure Server IDs", so assume that's the universe from which 14,714 certs have been revoked.

Question 1a

- Assume 200,000,000 users who will negotiate an SSL/TLS session with at least one of the 500,000 sites over the next two weeks.
- On average, how much bandwidth is VeriSign going to use per day distributing the RSASecureServer CRL?
  - You may assume user requests for CRLs are evenly distributed throughout the CRL's two-week validity period.

Question 1a

- 200M users, CRLs last 14 days, so on average 1/14th of the users will have to download the CRL each day.
- 200M/14 = 14.285714M downloads/day
- 515,243 bytes/download
- → ~7.360x10^{12} bytes of bandwidth <u>per day</u>

Question 1b

- Assume there also exists an OCSP responder for the same data
- If the average size of an OCSP request/response message pair is 3KB, how many OCSP responses would the average user have to request from the VeriSign OCSP responder per day in order to generate the same about of bandwidth usage as the CRL downloading you calculated in Question 1(a)?

Question 1b

- ~7.360x10^{12} bytes of bandwidth per day
- / 3KB/OCSP request/response pair
- ➔ 2.453x10^{9} OCSP round-trips
- / 200,000,000 users
- ➔ ~12.267 OCSP requests/user/day

Question 1c

- USG wants to issue a cert to each of 60 million passport holders.
- VeriSign is experiencing about a 3% revocation rate; assume that the same rate would apply for these certs.
- Approximately how big would the CRL be for the personal certs issued by the US Government?
  - You may assume that each CRL entry requires 35 bytes of storage when ASN.1 encoded.

Question 1c

- 60 million passport holders * 3% revocation rate → 1.8 million revoked certs at any one time.
- 1.8 million * 35 bytes/entry → 63×10^6 bytes in the CRL

Question 2

- Design a certificate enrollment protocol for enrolling each user for two certificates
  - Leverage the user's Kerberos credentials to authenticate the certificate requests to the CA.
  - You can choose whether users enroll for both signing and encryption certificates simultaneously (in one execution of the protocol) or sequentially (in two executions of the protocol).

Question 2

- Assume client generates all keys
  - Signature key pair: K_{Spub}, K_{Spriv}
  - Encryption key pair: K_{Epub}, K_{Epriv}
- The enrollment protocol has to provide:
  - Authentication of the client
  - Proof-of-possession of the corresponding private keys

Question 2 – Solution 1

- Client uses Kerberos to obtain a ticket for the CA. Assume the ticket contains shared secret $K_{C,CA}$.
- For each key, client forms a self-signed "certificate request" message (e.g. PKCS#10) that contains the public key and identifying information
  - CertReqS = $\{K_{Spub}, \text{Username}\}K_{SPriv}$
  - CertReqE = $\{K_{Epub}, \text{Username}\}K_{Epriv}$
- *Only works if K_{Epriv} can also sign!*

Question 2 – Solution 1

- Client sends the cert requests to the CA encrypted with the Kerberos shared secret
- C→CA: {CertReqS, CertReqE}$K_{C,CA}$
- CA decrypts the message (which authenticates that it came from C)
- CA verifies the signatures on CertReqS and CertReqE, yielding proof-of-possession of the corresponding private keys

Question 2 – Solution 1

- CA compares the username inside the requests with the identity associated with the Kerberos key $K_{C,CA}$
- CA issues certs CertS and CertE binding the keys to the username (or whatever identity information he wants to be in the certs).
  - CertS = $\{K_{SPub}, \text{username}\}K_{CApriv}$
  - CertE = $\{K_{EPub}, \text{username}\}K_{CApriv}$
- CA sends the certs back to the client (unencrypted).

Question 1 – Solution 2

- What if you can't sign with the encryption key?
  - If C can only encrypt, how do you do proof-of-possession?
  - Method 1: add a challenge response round (but that adds round-trips)
  - Method 2: encrypt the cert in the reply

Question 2 – Solution 2

- Client uses Kerberos to obtain a ticket for the CA. Assume the ticket contains shared secret $K_{C,CA}$.
- For each key, client forms a "certificate request" message (e.g. PKCS#10) that contains the public key and identifying information. Only CertReqS is signed.
  - CertReqS = $\{K_{Spub}, \text{Username}\}K_{SPriv}$
  - CertReqE = $\{K_{Epub}, \text{Username}\}$ unsigned
    - Could also sign with K_{SPriv}

Question 2 – Solution 2

- Client sends the cert requests to the CA encrypted with the Kerberos shared secret
- C➔CA: {CertReqS, CertReqE}$K_{C,CA}$
- CA decrypts the message (which authenticates that it came from C)
- CA verifies the signatures on CertReqS, yielding proof-of-possession for the signature key

Question 2 – Solution 2

- CA verifies identity info
- CA issues certs CertS and CertE
  - CertS = $\{K_{SPub}, \text{username}\}K_{CApriv}$
  - CertE = $\{K_{EPub}, \text{username}\}K_{CApriv}$
- CA sends the certs back to the client; CertS can go unencrypted but at least CertE is encrypted to K_{EPub}
  - CA→C: CertS, $\{\text{CertE}\}K_{EPub}$
- Client has to decrypt with K_{EPriv} to obtain CertE, thus proving possession in order to use the cert.

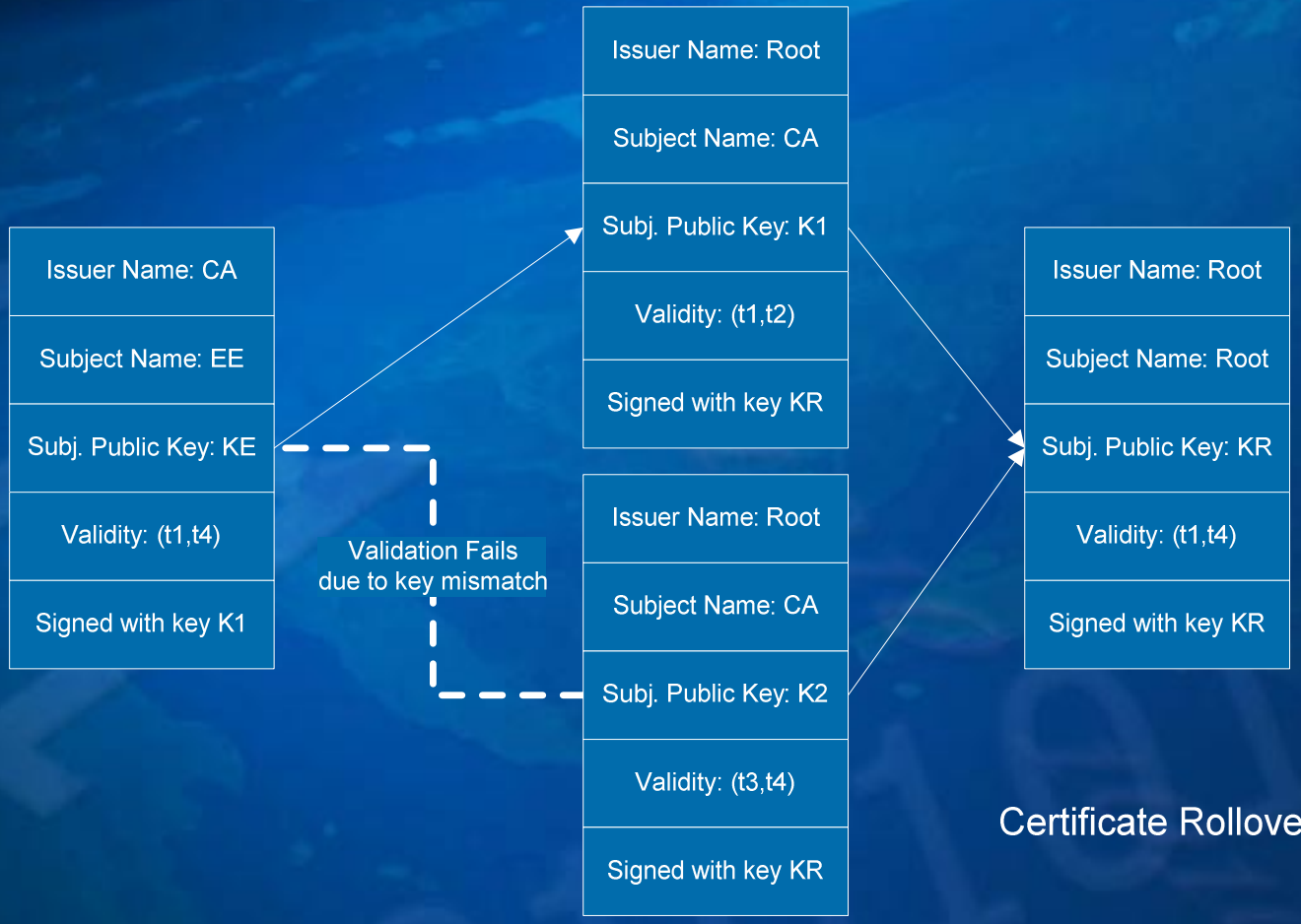
Question 3

- Modify the protocol you design in Question 2 to include a key escrow feature for the encryption key pair.

Question 3

- With client-side key gen, just need to send the encryption private key along with the encryption cert request
- C→CA: {CertReqS, K_{EPriv}, CertReqE}$K_{C,CA}$
- CA verifies that K_{EPriv} and K_{EPub} (in CertReqE) match
  - Only issues CertE if they verify
  - No additional POP required, since the server sees the private key

Question 4 – Cert Rollover

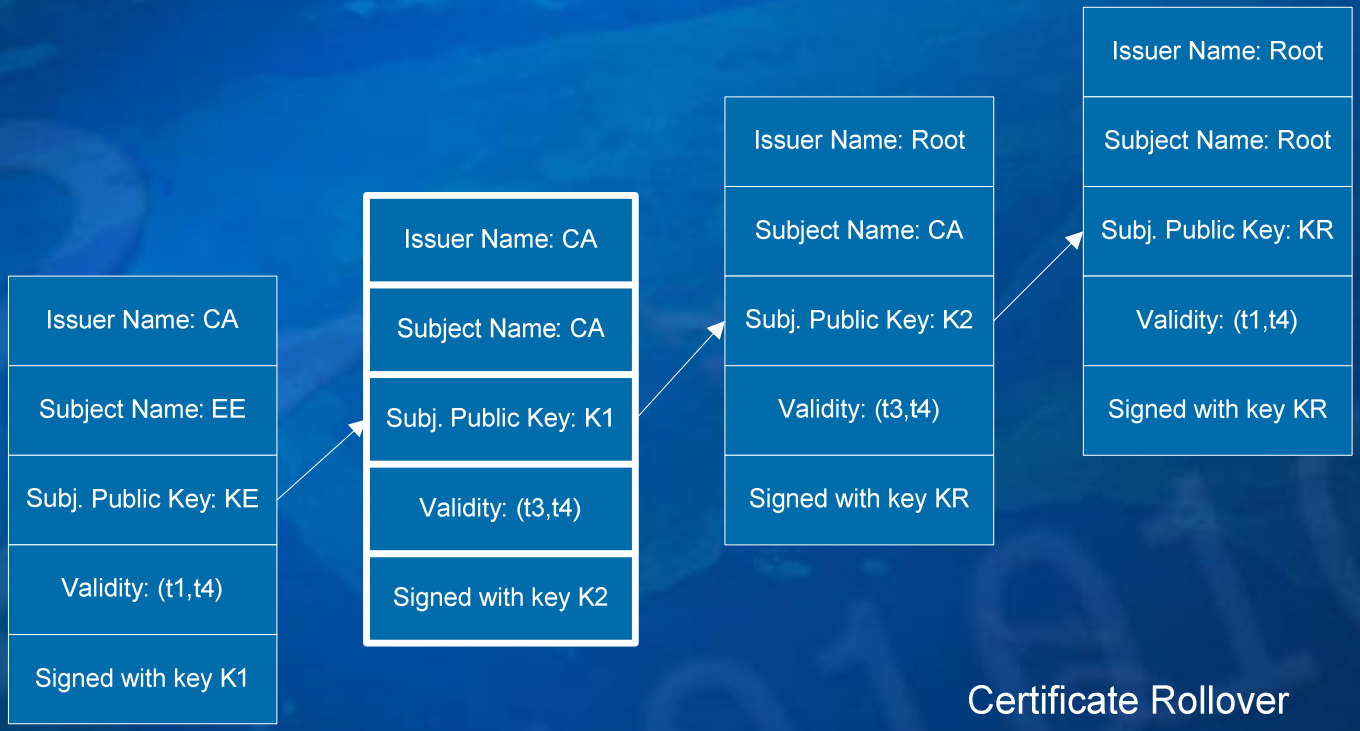


Certificate Rollover

Question 4a

- Assume that t1 < t2 < t3 < t4. Make the end-entity certificates validate at times t3 < t < t4 without re-issuing.

Question 4a

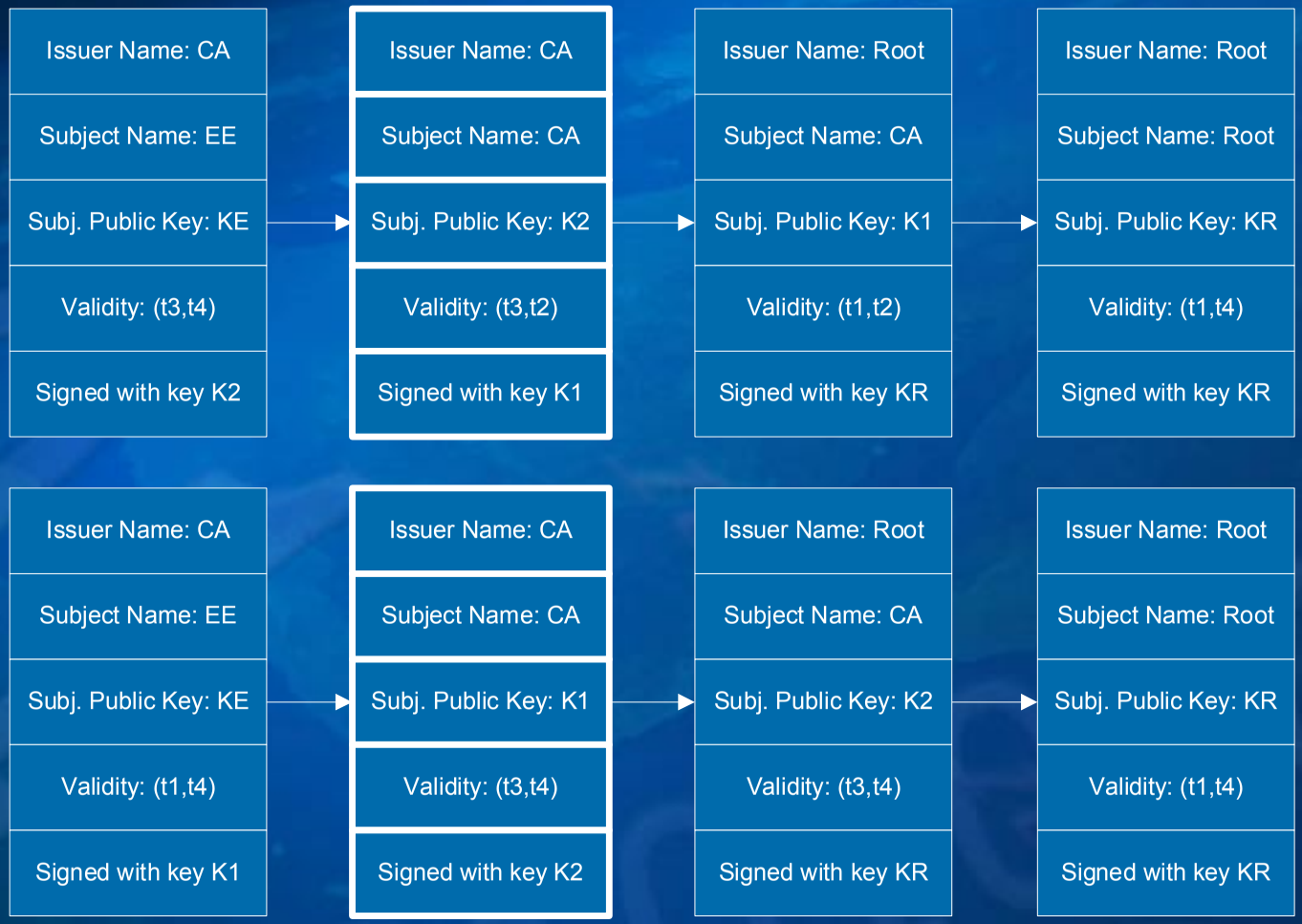


Certificate Rollover

Question 4b

- Assume that t1 < t3 < t2 < t4. For the period of time t3 < t < t2 end entity certificate should be able to chain-validate under both the old and new intermediate certificates.

Question 4b


Issuer Name: CA
Subject Name: EE
Subj. Public Key: KE
Validity: (t3,t4)
Signed with key K2
Issuer Name: CA
Subject Name: CA
Subj. Public Key: K2
Validity: (t3,t2)
Signed with key K1
Issuer Name: Root
Subject Name: CA
Subj. Public Key: K1
Validity: (t1,t2)
Signed with key KR
Issuer Name: Root
Subject Name: Root
Subj. Public Key: KR
Validity: (t1,t4)
Signed with key KR
Issuer Name: CA
Subject Name: EE
Subj. Public Key: KE
Validity: (t1,t4)
Signed with key K1
Issuer Name: CA
Subject Name: CA
Subj. Public Key: K1
Validity: (t3,t4)
Signed with key K2
Issuer Name: Root
Subject Name: CA
Subj. Public Key: K2
Validity: (t3,t4)
Signed with key KR
Issuer Name: Root
Subject Name: Root
Subj. Public Key: KR
Validity: (t1,t4)
Signed with key KR