



IST-2001-34340

D8.4 Audit Code Report

Distribution List :	Project Partners
Author :	Christophe DELAHAYE, AQL
Distribution List :	Project Partners
Authorised by :	
Date of Issue :	29 April 2004
Issue :	1.1
File Name :	WP8-D8.4-1.1.doc
Work Package :	WP8 Homologation issues and security guidelines
Deliverable Number :	D 8.4
Deliverable Type :	Final
Deliverable Nature :	
Total Number of Pages :	14
Contact Details for EUROPEPKI :	Project Coordinator mail : web site : www.europepki.org

0 Table Of Contents

0	Table Of Contents	2
1	Document Control	3
1.1	Abstract	3
2	Introduction and Glossary	4
2.1	Glossary	4
3	Audit synthesis.....	5
3.1	Review Steps.....	5
3.2	Review Scope and Main Results	5
4	Notes About The Specifications	7
5	Access Control Modules (numbers RA-15 and CA-15).....	8
6	The Key Management Modules.....	9
6.1	The Key Ceremony Modules.....	9
6.1.1	The Core API: org.eupki.kc.kc.KCAPIImplementation	9
6.1.2	The Class org.eupki.kc.CertRequestStructure	9
6.1.3	Other Classes.....	10
6.2	The Key Generation and Storage Modules	10
7	The CA API module (module number CA-10).....	11
7.1	The RAResultProcessorMDB Class.....	11
7.2	The Other Classes.....	12
8	A Few Global Notes	14

1 Document Control

<i>Issue</i>	<i>Date of Issue</i>	<i>Comments</i>
0.1	05 January 2004	document creation.
1.0	12 March 2004	Final Version
1.1	26 April 2004	Minor form correction on the final version

1.1 Abstract

AQL performed an audit of a part of source code on December 2003, this document is the reports the results of this audit.

2 Introduction and Glossary

The section 4 sums up the results of the audit. Each following sections provides the details for each module: documentation, access control, key management and the CA module itself.

2.1 Glossary

OCSF	On-line Certificate Status Protocol
PKI	Public Key Infrastructure
CA	Certification Authority
RA	Registration Authority
CRPKI	Cryptographic Resource for Public Key Infrastructure

3 Audit synthesis

3.1 Review Steps

The review started with the study of the specifications and the documentation included in the source tree (`doc/*.txt`). The "javadoc" documentation was partially built in order to get an overview of the applications (It is unfortunate that the provided Ant file is not usable for that without any installation of JBoss.). Then a sample of the source code was chosen and studied.

The source tree was checked out on 15/12/2003 at about 10:45. As the announce from Eduart Tric of the 9th source delivery has been received during the afternoon of the same day, it is possible that the reviewed source code is not the latest revision. So the recipients of this report should check whether the errors reported here are fixed in the current revision at the time of reading.

3.2 Review Scope and Main Results

Although the package "org.eupki.common.ext" is part of the CA core module, it was not studied because it is dedicated to encoding and decoding. These operations are not really security relevant and interoperability testing is a good support for validation.

Table 1. Studied source files

Filename, the module and the common package prefix being removed	Revision number	Length in lines, all comments included
<code>common/ac/AccessControl.java</code>	1.12	397
<code>ra/ejb/services/OperatorProfileImpl.java</code>	1.12	305
<code>ca/EuPKICertificateFactory.java</code>	1.5	590
<code>ca/comm/RAResponseProcessorMDB.java</code>	1.15	860
<code>ca/comm/RAResponseValidator.java</code>	1.15	504
<code>ca/comm/RequestReceiverMDB.java</code>	1.15	221
<code>ca/ejb/services/CRLFactory.java</code>	1.15	844
<code>kgs/ejb/services/KGSAdminAPIBean.java</code>	1.3	1194
<code>kc/KCAPIImplementation.java</code>	1.4	587
<code>kc/KeysGenResponse.java</code>	1.4	109
<code>kc/SoftwareKeysGenResponse.java</code>	1.4	41

Filename, the module and the common package prefix being removed	Revision number	Length in lines, all comments included
kc/CertRequestStructure.java	1.4	558
kc/CommandLineRunner.java	1.2	870
Total: 13 files		Total: 7080 lines

4 errors were discovered: one in the initialisation of the random number generator, another one in the CRL management which is incomplete, a third one in error logging and the last one in the assembly of the CRL issuer DN. There are detailed below.

In this document, fully qualified routine names are spelt as in C++: *qualified class name::method name*. When the considered routine is "static" the Java form (dot separated package or class names then the method name) may also be used. The pathnames are relative to the top of CVS working directory.

4 Notes About The Specifications

The specifications used are the latest documents available on 2003-12-11.

WP4-D4.1 states that almost all messages must be authenticated at SSL level. In such a case, the protocol stack is HTTP over SSL over TCP. However, WP4-D4.2 states that XML-sig must be used. It is not clear which mechanism must be used (using both is useless). The choice of SSL implies that a CA is required before the web applications start.

The structure of the database tables is specified in the document WP4-D4.3. The script `ra/src/sql/log.sql` creates some tables. The reviewer noticed that the specifications do not mention the audit tables created by the script. The missing tables are named "AuditLog", "AccessLog" and "ErrorLog". Moreover, the document WP4-D4.1 mentions access rights (in modules RA-15, CA-15, KGS-15) that do not appear in the tables specifications.

5 Access Control Modules (numbers RA-15 and CA-15)

In the source code sample, the reviewer did not find any access control to the database. If an intruder may have access to the network, (the requirement for requests and replies authentication suggests that), it might also write into the database, for instance he might grant to himself full rights.

For these modules the reviewer studied the following classes:

```
org.eupki.common.ac.AccessControl,      org.eupki.common.ac.OperatorProfile,  
org.eupki.ra.ejb.services.AccessControlBean,  
org.eupki.ra.ejb.services.OperatorProfileImpl,  
org.eupki.ra.ejb.interfaces.OperatorData.
```

The routine `org.eupki.common.ac.AccessControl::authenticate` converts all the exceptions into an `InvalidCertificateException`, including standards ones. This is likely error prone for the client.

`org.eupki.ra.ejb.interfaces.OperatorData` is expected to be equivalent to the table OPERATOR described in the § 2.1.13 of the document WP4-D4.3. However, the field BO_ID in this table does not occur in the class definition. And the class definition has an additional field named "isAdmin". This field was expected to be defined from the database content, particularly the "ROLE_ID" table attribute but is instead provided by the class client.

In the constructor of `org.eupki.ra.ejb.services.OperatorProfileImpl`, if no administrator is defined, a new one is **still** automatically created by the routine `findOperatorByCertificate`. This piece of code is marked as to be removed although `doc/modules.xls` suggests that this class is considered complete. About the efficiency of `findOperatorByCertificate`, the reviewer noticed that the given certificate that is looked for in the database might be DER encoded for each comparison instead of once before the search loop.

6 The Key Management Modules

6.1 The Key Ceremony Modules

6.1.1 *The Core API: org.eupki.kc.kc.KCAPImplementation*

This class includes the implementation of the modules the number of which are KC-26, KC-30 to KC-32.

About the routine `generateSoftwareKeys`, the reviewer guessed that the loop beginning with the comment "form the secret", extending over the lines 118 to 130 must append all the secrets from the input collection and noticed that this loop does not have any effect when the input has only one element. This is likely a bug. Moreover, this part can be written more simply and possibly more efficiently by copying the content of the input collection into a `java.io.ByteArrayInputStream`. This may be more efficient thanks to a reduced number of array allocations and copies.

About the routine `generateCAPKCS12SignedCertificate` of the same class, at line 216 to 220, there is a test of the validity of the serial number of the certificate being generated. This test rejects only the serial number 0 although it should also reject serial number 1 because 1 is used for the CA certificate itself. (See `generateSoftwareSelfSignedCertificate` in the same class.) Please note however that this serial number is an elapsed time expressed in milliseconds since the CA certificate creation, so 0 and 1 are very unlikely.

`exportPKCS12` uses 2 different passwords, one to encrypt the private key and another one, likely used for integrity. It may be useful to notice about this that some applications (mailer or browsers) are said unable to handle such files.

In `printSecret`, the handling of the default printer is quite unusual: the print request is handled by a loop the body of which is always executed only once.

6.1.2 *The Class org.eupki.kc.CertRequestStructure*

`CertRequestStructure` is used to record the components of a X509 certificate. Its method aimed to set the version number accepts only 1 and 3. Although these are the right version numbers of the specifications, the reviewer is not sure that the author accounts that the *encoding of the version number is one less than the actual specification number*, so the encoded certificate may be wrong. As this class does not include the routine in charge of the DER encoding, the reviewer can not check this. About the encoding, see for instance the [RFC 3280] entitled *Internet X.509 Public Key Infrastructure - Certificate and Certificate Revocation List (CRL) Profile*.

The reviewer noticed that error processing in this class is different. In all the other classes checked, when an argument is checked and stated invalid, the

method throws an exception the type of which provide some information about the error. In the methods of `CertRequestStructure` related to X509 extensions processing, when an argument is invalid, the method simply ignores it, it does nothing.

The description of `getNotBefore` indicates that not having a start date is valid and that means that the current date (likely at signing time) must be used instead. However, the method `isValid` returns false in such a case.

The comments of the methods `getNotAfter` and `getNotBefore` are swapped.

6.1.3 *Other Classes*

The class `org.eupki.kc.KeysGenResponse` is a simple structure that encapsulates standard half keys, represented by classes from `java.security.*.SoftwareKeysGenResponse`, an heir of the first one, is almost fully the same, as it does not add any features (nether attributes nor methods). So the reviewer does not think about a reason of the existence of the second class.

The parsing of the command line options in `org.eupki.kc.CommandLineRunner` is quite redundant. As the handling of some options is common in both kind of command, (at least "o" for the output directory, "i" for the properties file and "v" for the indication of what is being done), this worth having a dedicated method for them instead of duplicating the code chunk in both "switch".

Adding certificate subject alternate names extensions in `CommandLineRunner::createEntity` is also done by duplicating chunk of code and changing one literal string and a symbolic constant. Since this method is error prone, (see for instance the bug in the CRL issuer DN initialisation), this is better done as a loop using an array of string and an array of integer for the variable part (alternatively with string parameters a `java.util.HashMap` may be used. The same tip also applies to the key usage extension in the same routine.

6.2 The Key Generation and Storage Modules

The source file `kgs/src/main/org/eupki/kgs/ejb/services/KGSAdminAPIBean.java` is large but contains large chunk of commented out code. The disabled methods are mostly some certificate and key queries.

This class includes some security sensitive update in the databases, particularly it manages administrators rights; but the reviewer did not found in this class neither authentication nor access-control checks. The user identifier is only used for logging purpose. It is not even passed to the methods in charge of the actual database queries and update, so access control can not be delegated to the databases oriented classes.

7 The CA API module (module number CA-10)

The checked source files for this module includes the classes `RequestReceiverMDB`, `RAResultProcessorMDB`, `RAResultValidator`, from the package `"org.eupki.ca.comm"`, `org.eupki.ca.EuPKICertificateFactory` and `org.eupki.ca.ejb.services.CRLFactory`. The revision number of all these files is 1.15.

7.1 The RAResultProcessorMDB Class

This class has an unused global field the name of which is `serverKeyStore`. If it is a file path, it is such Unix centric that the class would not function properly under Windows™.

The procedure `onMessage` reads, at lines 293 to 332, a big loop the body of which, never run in its entirety even once, is mainly a sequence of test of the dynamic type of the routine argument. This kind of code is what the object oriented languages are aimed to eliminate thanks to polymorphism and dynamic binding. Indeed, in a good object oriented design, the usual way of doing this kind of thing is using a polymorphic factory method to create `repWait`. For the current chunk of code, such a method would likely be based on `java.lang.Object.clone` and possibly some update of the state. A comment indicates the author wants to avoid a chain of "if" constructs but the usual problem with such chain (large shift to right due to indentation) is not applicable here. Moreover this false loop prints some debug information to the process standard output, which may not exist in a servlet¹, the author should use the dedicated logger as this is done in the following lines.

In the same procedure, there is a big "switch" construct extending from the line 360 to the line 375. The selector of the construct is the incoming message type. This is another example of large explicit choices that object oriented languages are intended to avoid. However, in this case there is only one apparent type: the class `org.eupki.common.comm.cmp.CMPMsgHeader`, so using polymorphism here would consist to introduce a class hierarchy with at least one routine, say `processMessage` the body of which is, roughly speaking, the statement of each "case" of this "switch". As it is likely that other processing of this type of message have this thing of variations, the reviewer thinks that doing so would be useful. Another example of such kind of processing is the validation of a request from a RA to a CA in `org.eupki.ca.comm.RAResultValidator::validateRequest`.

Again in the same long procedure, the lines 389 to 396 create a request for the generation of a key pair. As a comment states it, the request should not be hard coded to RSA 1024: the acceptable size depends on, among other things, the lifetime of the key, its future usage (encryption or signature) and the laws of the country of the user.

¹ Some servlet engine, for instance JRun, discard data sent to the standard output.

Some parts of the software represent certificate serial numbers as `java.lang.Long` and other ones use `java.math.BigInteger`. For instance the procedure `onMessage` uses primitive integers for revocation requests and `BigInteger` for suspension requests.

In the revocation processing part of this class, the reviewer did not find any update of the certificate status in the CA databases: the objects used to record the revocation are all local, so the update can not be made persistent. Moreover `org.eupki.ca.EuPKICertificateFactory` does not have any routine for database update as it has for certificate creation or certificate renewal. A comment in the code of `onMessage` ("`// FIXME: add to CRL (when CRL support is ready)`" at line 489) supports the statement. However `doc/modules.xls` states that this class is complete.

7.2 The Other Classes

A long part of `org.eupki.ca.comm.RARequestValidator::validateRequest` checks that the X509 extension set of the current certificate generation request equals the set of extension specified in the certificate profile. Since the extension set from the profile is retrieved as a `java.util.Set`, this could be done in a simpler way and more efficiently by filling in a set with the request extensions and replacing the 2 large nested loops by `Set::equals`. Using a `java.util.HashSet` would provide a cost in time roughly linear.

The same procedure also must check that a revocation request from a RA is valid. In this processing, -1 is used as a serial number, which is not compliant with the usual rules: these ones states that a serial number must be positive. This is the case for instance of the X509 profile for Internet known as PKIX. (See the [RFC 3280] or its previous version, [RFC 2459] for details.)

`onMessage` from `org.eupki.ca.comm.RequestReceiverMDB` reads "]" from " + `pMessage.getJMSDestination()`. So the error logged indicates the recipient address instead of the sender one.

The class `org.eupki.ca.EuPKICertificateFactory` is only formed of "static" fields and routines. That implies that a given process, likely a running JVM or strictly speaking, one class loader in a running JVM, can support only one CA.

In `rg.eupki.ca.EuPKICertificateFactory.databaseSync`, the freshly created certificate is stored in the database and then in a file the name of which is hard-coded to `gigi.der`. This file does not have any obvious usefulness: it is not directly used after its creation and its content may be unreliable if the application is multi-threaded and because message processing is asynchronous. The reviewer thinks that this is an old chunk of code that used to be used for debugging purpose.

The method `org.eupki.ca.ejb.services.CRLFactory::buildX509Hash` must build a table of the fields forming an X500 DN provided by the given certificate part. In the last step of the filling in, the value for the "O" key of this table

is initialised with the "ST" field from the DN. So "ST" appears twice: as "ST" and as "O".

In the method `buildNameDetails` from the same class, the reviewer noticed that the "O" field of the given DN is ignored. As it is used in other routines (see for instance `buildX509Hash` above), that may cause a failure during LDAP searches. So this is likely a new bug.

The method `scanDatabase` of the same class is in charge of building the CRL and updating the certificate directory. In this routine, the certificate status "expired" is not processed, a comment suggests that this case be handled in a previous database scan. However, nothing in the code of this class supports this statement. This case may be handled somewhere else. For instance, an expired certificate might be removed from the directory by the routine which marks it as revoked but this would be an error prone design since the other cases or certificate withdrawn (namely revocation and suspension) are handled by `scanDatabase`. The reviewer suggests that the process of expired certificate withdrawn should be checked against this case.

8 A Few Global Notes

The reviewer thinks that the core functions are not easy to find because a large part of the code is related to communications and inter-modules messages control hiding the core functions.

About the coding style, checking that the given string are not empty is often written as `var.length() <= 0`. That seems a bit strange since string length can not be negative, this is more often coded as `var.length() != 0`.