APM

POSEIDON HOUSE • CASTLE PARK • CAMBRIDGE CB3 0RD UNITED KINGDOM
+44 1223 515010 • Fax +44 1223 359779 • Email: apm@ansa.co.uk • URL: http://www.ansa.co.uk

ANSA

SSL Package Comparison

Takanori Ugai

Abstract

We have investigated some SSL implementation in Java to use the secure communication in FlexiNet and MOW. This is a comparison report of several SSL implementation. Functional comparison is done by public information through web and documents and performance comparison between IAIK and JCP is done to implement FlexiNet to use them.

<Doc No. (Sum Info)> **Approved** Technical Report 23 March 1998

Distribution:
Supersedes:
Superseded by:

TABLE OF CONTENTS

1 SSL PACKAGES IN JAVA

We are investigating the integration of FlexiNet and SSL for secure communication in the FollowMe Mobile Object Workbench.

All SSL packages here are available outside the U.S and could be used in Java except Cryptix, which only provides the JCE implementation.

- ♦ IAIK SSL (http://jcewww.iaik.tu-graz.ac.at/)
  - Price :
    - IAIK SSL Developers Licence with IAIK JCE $430 (400EURO)
    - additional license up to 10 $120 (110EURO)
  - Entirely written in Java
  - Provides java.net.Socket interface, which means it is easy to integrate with RMI or other systems
  - has some APIs to enforce the authorisation policy
  - has an API to add encryption algorithms
  - Enough stable for proto-type systems
  - There is an active mailing list for users
  - The response to queries by e-mail is quite good
  - Based on JCE standard interface
- ♦ JCP (http://www.jcp.co.uk/)
  - Price :
    - Research License 1000 pounds
    - Commercial license   not-specified bt around 10k pounds
    - Compulsory support license 1000 pounds additional
  - Entirely written in Java
  - Provides java.net.Socket interface
  - has some APIs to enforce the authorisation policy
  - has an API to add encryption algorithms
  - Based on JCE standard interface
- ♦ J/SSL (http://www.baltimore.ie/)
  - Entirely written in Java

  - Provides java.net.Socket interface
  - has some APIs to enforce the authorisation policy
  - has an API to add encryption algorithms
  - Based on JCE standard interface
- SSLeay
  - http://psych.psy.uq.oz.au/~ftp/Crypto/
  - http://noc.kpnw.org/~scott/ (Java implementation)
  - free even for commercial purposes
  - not in Java but native code + Java interface
  - only available on Solaris and NT
  - requires of modification of JDK sources
  - The client information is not being passed back up to the application
- Cryptix (http://www.systemics.com/software/)
  - Systemics is a Netherlands base company, but package is developed by volunteers
  - SSL is not released
  - free even for commercial purposes
  - entirely written in Java

2 COMPARISONS

2 .1 JCE Implementation

Cipher

	DES	IDEA	RC4	RC2	3DES	RSA	Blowfish	SPEED	CAST
IAIK	y	y	y	y	y	y	n	n	n
JCP	y	y	y	n	y	y	n	n	n
J/SSL	y	n	y	n	y	y	n	n	n
IJCE	y	y	y	y	y	y	y	y	y

Hash function

	MD5	SHA0	SHA1	MD2	MD4
IAIK	y	y	y	n	n
JCP	y	y	y	n	n
J/SSL	y	y	y	n	n
IJCE	y	y	y	y	y

Signature

	MD5/RSA	RSA	SHA0/RSA	SHA1/RSA	DSA	MD2/RSA
IAIK	y	y	y	y	y	n
JCP	y	y	y	y	n	n
J/SSL	y	y	y	y	y	n
JICE	y	y	y	y	y	y

Mode for chipher (DES)

	CBC	ECB	OFB	CFB	PCBC
IAIK	y	y	y	y	y
JCP	y	y	y	y	n
J/SSL	y	y	y	y	n
JICE	y	y	y	y	y

Padding

	NULL	PKCS#1	PKCS#5	PKCS#7
IAIK	y	y	y	n
JCP	y	y	y	n
J/SSL	y	y	y	n
IJCE	y	y	y	y

JCE Compliant

	JCE 1.1	JCE 1.2
IAIK	y	y
JCP	y	n
J/SSL	y	n
IJCE	y	y

2 .2 ASN.1 and X509Certificate

Format : All of IAIK, JCP and J/SSL support Ascii (PEM) and binary (BER) format.

PKCS#7 (Certificate Chain Format) : IAIK and J/SSL support PKCS#7 but JCP not.

2 .3 SSL Functions

Protocols

	IAIK	JCP	J/SSL

RSA_WITH_DES_CBC_SHA	y	y	y
RSA_WITH_NULL_MD5	y	y	y
NULL_WITH_NULL_NULL	n	y	n
RSA_WITH_NULL_SHA	n	y	y
RSA_EXPORT_WITH_RC4_40_MD5	n	y	y
RSA_WITH_RC4_128_MD5	n	y	y
RSA_WITH_RC4_128_SHA	n	y	y
RSA_WITH_IDEA_CBC_SHA	y	y	n
RSA_EXPORT_WITH_DES40_CBC_SHA	y	y	n
RSA_WITH_3DES_EDE_CBC_SHA	y	y	y
RSA_WITH_RC4_MD5	y	n	n
RSA_WITH_RC4_SHA	y	n	n
RSA_EXPORT_WITH_RC2_CBC_40_MD5	y	n	n
DH_RSA_EXPORT_WITH_DES40_CBC_SHA	y	n	n
DH_RSA_WITH_DES_CBC_SHA	y	n	n
DH_RSA_WITH_3DES_EDE_CBC_SHA	y	n	n
DH_DSS_EXPORT_WITH_DES40_CBC_SHA	y	y	y
DH_DSS_WITH_DES_CBC_SHA	y	n	n
DH_DSS_WITH_3DES_EDE_CBC_SHA	y	n	n
DHE_RSA_EXPORT_WITH_DES40_CBC_SHA	y	n	y
DHE_RSA_WITH_DES_CBC_SHA	y	n	y
DHE_RSA_WITH_3DES_EDE_CBC_SHA	y	n	y
DHE_DSS_EXPORT_WITH_DES40_CBC_SHA	y	n	n
DHE_DSS_WITH_DES_CBC_SHA	y	n	n
DHE_DSS_WITH_3DES_EDE_CBC_SHA	y	n	n
DH_anon_EXPORT_WITH_RC4_40_MD5	y	n	y

DH_anon_WITH_RC4_128_MD5	n	n	y
DH_anon_WITH_3DES_EDE_CBC_SHA	y	n	y
DH_anon_WITH_DES_CBC_SHA	y	n	y
DH_anon_EXPORT_WITH_DES40_CBC_SHA	y	n	y
DH_anon_WITH_RC4_MD5	y	n	n

Accept Certificate

	IAIK	JCP	J/SSL
rsa_sign	y	y	y
dss_sign	y	y	y
rsa_fixed_dh	y	y	y
dss_fixed_dh	y	y	y
rsa_ephemeral_dh	y	y	y
dss_ephemeral_dh	y	y	y
fortezza_dme	y	y	n
fortezza_kea	n	n	y

Other functions

	Asynchronous handshake	proxy support	renegotiating	SunSSLAPIcomplient
IAIK	n	y	y	y
JCP	y	n	y	n
J/SSL	n	n	y	n

2 .4 Other issues

- Documentation quality
  - IAIK : poor

  - JCP : poor
  - J/SSL : rich (200pp manual)

- Impressions of APIs
  - JCP's and J/SSL's API are symmetric between the server side and the client side. IAIK's is not.
  - IAIK's and J/SSL's X509, ASN.1 Object API are more user friendly.

3 PERFORMANCE

3 .1 Cipher through SSL

We have obtained evaluation copies of IAIK SSL and JCP SSL and implemented FlexiNet to use them.

Time of call is the period to take a FlexiNet rpc call with no argument. Through puts are that small number is by small size packet by null call and the big number is by large packet by the call with 100kbyte string argument.

	TCP	IAIK(RSA_WITH_RC4_128_SHA)	IAIK(RSA_WITH_NULL_MD5)	JCP(RSA_WITH_RC4_128_SHA)	JCP(RSA_WITH_NULL_MD5)
call (msec/call)	8.1	158	130	168	147
through put (kbps)	500	33 - 66	48 - 110	37	47 - 110

3 .2 SSL negotiation

- IAIK

– RSA_WITH_RC4_SHA	335msec/connection
– RSA_WITH_NULL_MD5	335msec/connection

- JCP

– RSA_WITH_RC4_128_SHA	5035msec/connection
– RSA_WITH_NULL_MD5	4191msec/connection

Why is the JCP's handshake so slow?

- Because the asynchronous handshake makes a new thread.
- JCP SSL use java.security.SecureRandom, which sleeps a long time to generate a sophisticated random number for a session key and we could

not find a way to avoid this long sleep except replacing java.security.SecureRandom. IAIK SSL uses a own random number generator.

4 CONCLUSION

JCP is stable for production use and users can expect commercial level support.

IAIK has some small bugs but enough stable for productions and users can expect their support through the mailing list. IAIK seems to try to implement new futures like proxy support and PKCS standard.

Recommendation for ANSA research : IAIK is good enough for research purpose, but continue to man for JCP for commercial use.