JCOP31bio Technical Brief

***Overview**: This document contains a simple overview about the technical capabilities of the first biometry-enabled contact/contactless (dual-interface) member of the JCOP card OS family. Requests for further information may be directed at javacard@zurich.ibm.com.*

1. Basic specifications

JCOP is an IBM BlueZ implementation of the basic specifications [1] and [2] including refinements from Visa International set in the Visa OpenPlatform Card Implementation Requirements (http://www.visa.com/nt/suppliers/vendor). All necessary clarifications from ISO7816 and EMV 2000 are also incorporated into the implementation where so required by [1] and [2].

JCOP31bio is the first biometry-enabled dual-interface member of this family. It conforms to the VOP Card Implementation Requirements Configuration 3 with PK, Version 2.0 from February 2002.

2 Communications

2.1 Supported protocols

ISO7816 T=1 direct convention [default]*

ISO7816 T=0 direct convention*

ISO7816 T=1 inverse convention*

ISO7816 T=0 inverse convention*

ISO14443A T=CL

Note: The contact protocols of JCOP can be configured to support the clock-stop feature of certain terminals to save power consumption (typically done in mobile phones)

2.2 Supported speeds

2.2.1 Contact protocols:

At the default clock rate of 3.57 MHz, the following communication speeds can be attained:

9600 bit/sec [default]

19200 bit/sec

38400 bit/sec

57600 bit/sec

115200 bit/sec

2.2.2 Contactless protocol:

106 kbit/sec [default]

212 kbit/sec

424 kbit/sec

* The contact protocols of JCOP can be configured to support the clock-stop feature of certain terminals to save power consumption (typically done in mobile phones)

3 Memory availability for applications

3.1 EEPROM

3.1.1 Persistent Java heap

Used for allocating persistent objects, applets, and storage of post-issuance loaded applet code (aka packages).

Size: 13kB (without custom ROM applets and with one biometric algorithm in EEPROM).

3.1.2 Transaction buffer

Used to save data written transactionally, e.g. all persistent byte and short stores, as well as persistent parameters to Util.arrayCopy; see [1].

Size: 512 bytes

3.2 RAM

3.2.1 Transient Java heap

Used for allocating transient objects and arrays of type CLEAR_ON_RESET and CLEAR_ON_DESELECT.

Size: 523 bytes

3.2.2 APDU buffer

Used to hold incoming and outgoing communications data.

Size: 261 bytes

3.2.3 Java stack

Used to hold call parameters, local variables, and stack frames of the VM.

Size: 200 bytes

3.3 ROM

24kB free for applications*

* Custom applets may be submitted to Philips for inclusion into a ROM mask (see section 5)

4 Supported optional features

Certain features listed in [1] and [2] are not defined to be mandatory. The ones implemented in JCOP are listed below.

4.1 JavaCard

4.1.1 Garbage Collection

Fully implemented: Deleted objects, applets, and packages are fully reclaimed and the space can be used for other purposes after deletion.

4.1.2 Cryptographic Algorithms

JCOP31bio has the ability to generate RSA keys on the card. The following JavaCard API constants (see [1]) are implemented by this version of JCOP:

Ciphers:

ALG_DES_CBC_NOPAD

ALG_DES_CBC_ISO9797_M1

ALG_DES_CBC_ISO9797_M2

ALG_DES_ECB_NOPAD

ALG_DES_ECB_ISO9797_M1

ALG_DES_ECB_ISO9797_M2

ALG_RSA_PKCS1

ALG_RSA_NOPAD

Signatures:

ALG_DES_MAC8_NOPAD

ALG_DES_MAC8_ISO9797_M1

ALG_DES_MAC8_ISO9797_M2

ALG_RSA_SHA_ISO9796

ALG_RSA_SHA_PKCS1

ALG_RSA_MD5_PKCS1

MessageDigest:

ALG_SHA

ALG_MD5

RandomData:

ALG_SECURE_RANDOM

KeyTypes:

LENGTH_DES

LENGTH_DES3_2KEY

LENGTH_RSA_2048[1]

LENGTH_RSA_1024[1]

LENGTH_RSA_768[1]

LENGTH_RSA_512[1]

TYPE_DES_TRANSIENT_RESET

TYPE_DES_TRANSIENT_DESELECT

TYPE_DES

TYPE_RSA_PUBLIC

TYPE_RSA_PRIVATE[2]

TYPE_RSA_CRT_PRIVATE

KeyPair:

ALG_RSA_CRT

4.1.3 APDU class

The method APDU.getProtocol() returns according to [1] the currently activated communications protocol. In compliance with [1], JCOP31bio returns APDU.PROTOCOL_T0 (0) if T=0 is running, and APDU.PROTOCOL_T1 (1) if T=1 is running. JCOP31bio returns none of these constants if T=CL is running. Hence, using a query of the form

```
if ((APDU.getProtocol() != APDU.PROTOCOL_T0) && (APDU.getProtocol() != APDU.PROTOCOL_T1))
```

can be used to cease computation in an applet that does not wish to execute if run over the contactless interface.

4.2 OpenPlatform

4.2.1 Global PIN

Fully implemented: All described APDU and API interfaces for this feature are present.

[1] All multiples of 32 (bit) are supported as valid RSA key lengths. Thus, key length values such as 736 (bits) can be passed as parameters to the respective functions.

[2] Private Keys must be loaded with key material. On-card key generation is only supported for RSA keys in CRT format.

4.2.2 Multiple SecurityDomains / DAP

JCOP31bio allows the installation of multiple Security Domains as well as Mandated DAP verification.

4.3 Biometry

4.3.1 Scope

JCOP31bio implements the full biometry API as defined in [3].

4.3.2 Supported constants

Different on-card biometric match algorithms can be made available for JCOP31bio. The first one supported is a fingerprint matching algorithm. The supported API constant in the API thus is

org.javacardforum.javacard.biometry.BioBuilder

FINGERPRINT

5 Supported Hardware

The supported configuration includes a 1kB Mifare Emulation mode (“Mifare Standard”) ensuring interoperability of JCOP31bio in existing Mifare infrastructures. The usual JCOP ROM applet integration facilities can be used to create custom masks*.

5.1 Philips P8RF5016

64 kB ROM → 24kB free for ROM’d applets in Custom Mask Process

16 kB EEPROM

2300 Bytes RAM

Triple-DES coprocessor

FameX RSA coprocessor

Mifare Standard Emulation

* Custom applets may be submitted to Philips for inclusion into a ROM mask. The maximum package size is 16kB.

6 Performance figures

In the absence of standard performance tests, typical applet's operations are timed. The protocol used is T=1 at 9600 bit/sec. The reader clocks the chip at 4 MHz. The applets are the Visa approved versions after having been initialized and populated with keys as required for Visa testing. To avoid measuring communications overhead, timing is measured between the last APDU byte sent to the reader and the first byte returned from the card.

Operation	ETUs	msec
SELECT CardManager[1]	79	7.3
INIT UPDATE CardManager[1]	293	27.2
EXTERNAL AUTH CardManager[1]	187	17.4
Install VisaCash[2]	3868	359.7
SELECT VisaCash[1]	88	8.8
Initialize LOAD for VisaCash[1]	438	40.7
Perform LOAD for VisaCash[1]	896	83.3
Initialize DEBIT for VisaCash[1]	122	11.3
Perform DEBIT for VisaCash[1]	984	91.5
ReadBalance from VisaCash[1]	58	5.4
SELECT VSDC[1]	102	9.5
GenerateAC from VSDC[1]	1436	133.5

The PK operations are largely dominated by the hardware speed of the Philips FameX PK coprocessor. Note that on-card key generation is a random-based process; thus the figure given is only an average value. Values are measured at low-level; depending on Java programming skills, application-level code can add some time to the values depicted here. Also note that two figures are given: The first for typical contact-less performance (slower, requiring less power consumption during contactless operation), one for typical contact-only operation (more power, hence faster).

Operation	Msec
1024 bit CRT public key operation (F4)	24 / 18
1024 bit CRT private key operation	309 / 163
Generate 1024bit CRT key	~7900 / ~4200
Generate 2048bit CRT key	~83000 / ~47000

[1] Second time command is executed (to eliminate potential applet setup effects)
[2] On 'clean' card (to eliminate potential EEPROM clearing effects)

A Revision History

1.0 Initial document

1.1 Specification amendment on RSA keys (4.1.2), Free ROM applet size clarified (5)

1.2 Added comment on clock-stop feature (2.1)

B References

[1] Sun Microsystems: JavaCard 2.1.1 http://java.sun.com/products/javacard

[2] Global Platform Consortium: OpenPlatform 2.0.1' http://www.globalplatform.org/

[3] JavaCard Forum: Biometry API specification: http://www.javacardforum.org/Documents/Biometry/biometry.html