FUJ00001379

Document Title: **Generalised API for OPS/TMS**
Appendix B: Cryptography and Key Management

Document Type: Technical Design Standard

Release: N/A

Abstract: This appendix provides an overview of the Key Management Service developed by ICL Pathway for the release CSR+. It defines the cryptographic interfaces that have been developed to support the applications available at this release.

The main document provides the information required to plan the development of new applications and describes in more detail the architecture set out in the OPS Architecture Specification.

Both documents are supplied under the terms of the Codified Agreement to POCL to facilitate the procurement of applications to run on the Service Infrastructure (interfacing with OPS and TMS).

This document is only available to organisations outside ICL Pathway through formal Non-Disclosure Agreement.

Document Status: APPROVED

Author & Dept: Patricia Morris, Technical Design Authority

Contributors: David Johns, Janet Dore

Reviewed By: ICL Pathway: Terry Austin, John Dicks, David Johns,

Dave Tanner, Geoffrey Vane, Peter Wiles

POCL: Bob Booth

Comments By:

Comments To: Document Controller & Author

Distribution: ICL Pathway Library and Reviewers

FUJ00001379

0.0 Document Control

0.1 Document History

Version No.	Date	Reason for Issue	Associated CP/PinICL No.
0.10	11/01/00	Initial version for review	
0.10a	20/01/00	Internally reviewed	
0.10b	21/01/00	Internally reviewed	
0.10c	27/01/00	Internally reviewed	
0.11	28/01/00	Reviewed by POCL	
0.12	18/02/00	For review by POCL	
0.13	06/03/00	For internal review	
0.14	14/03/00	For review	
1.0	30/03/00	Approved	

0.2 Approval Authorities

Name	Position	Signature	Date
T. Austin	Development Director		
J. Dicks	Customer Requirements Director		
R. Booth	POCL		

0.3 Associated Documents

Reference	Version	Date	Title	Source
TD/ARC/030	0.4	12/11/99	OPS Architecture Specification	ICL Pathway
TD/ARC/029	0.4	12/11/99	TMS Architecture Specification	ICL Pathway

FUJ00001379

0.4 Abbreviations/Definitions

Abbreviation	Definition
ACF	Auto-Configuration
AP	Automated Payments
API	Application Programming Interface
APS	Automated Payment Service: counter application supported by Horizon.
AUDS	Audit Server
base-64 encoding	An encoding that uses characters from the set "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz 0123456789+/"
C	A UNIX-derived programming language
C++	Object oriented version of C
CM	Configuration Management
Crypto API	Cryptographic Functions Application Programming Interface
DLL	Dynamic Link Library
DSA	Digital Signature Algorithm
FAD	Financial Accounting Division
FEK	Filestore Encryption Key
FTMS	File Transfer Managed Service
KMA	Key Management Application
KMS	Key Management Service
L&G	Landis and Gyr
OBCS	Order Book Control Service; counter application supported by Horizon, which supports a similarly named DSS application.
OPS	Office Platform Service. The provision and support of the hardware and software at Outlets including the Desktop environment of the Horizon system.
OSD	Office Supply Division
PMMC	PostMaster's Memory Card
PO	Post Office
POCL	Post Office Counters Ltd
POLO	Post Office Logon

FUJ00001379

Rambutan	A symmetric encryption algorithm implemented in Zergo communications hardware.
RIPOSTE	Retail Integrated Point Of Sale system in a Transaction Environment: product from Escher that provides both the infrastructure and the Desktop environment of the Horizon system. The definitions in this manual refer to version 6.0 onwards.
RPC	Remote Procedure Call
SHA	Secure Hash Algorithm
SI	Software Issue
TIP	Transaction Information Processing: POCL application that handles transaction data returned from Horizon.
TMS	Transaction Management Service. The hardware and software required for the replication, transmission and management of transactions committed to the Horizon Riposte Message Store and Pathway Data Centres, or vice versa.
TPS	Transaction Processing System: application that collects transaction information and returns it to TIP.
VPN	Virtual Private Network

0.5 Changes in this Version

Version	Changes
1.0	Includes changes arising from POCL review of V0.14

0.6 Changes Expected

Changes
This document reflects the current implementation. The provided descriptions and definitions may be subject to change control as determined by technical and/or operational needs.

FUJ00001379

0.7 Table of Contents

FUJ00001379

B.1 Scope

The information in this appendix is provided to enable the overall planning of new applications using cryptographic functionality. It gives an overview of the cryptographic facilities provided by the ICL Pathway Key Management Service (KMS) and identifies those interfaces available to new applications that interface to the OPS and TMS. It does not specify the information necessary to enable the detailed design and implementation to proceed or discuss implications of its use beyond the scope of OPS and TMS.

Any development that interfaces with the ICL Pathway system may impact the security of existing applications and so must be subject to a *Security Evaluation Review* process by ICL Pathway. See Appendix C, *System Management,* for further information.

Any usage by new applications of the cryptographic interfaces will require the impact on existing domains, or of new domains, to be scoped by ICL Pathway. For example, even use of an existing key by a new application could have an impact and so must be evaluated.

B.2 Introduction

Security within TMS and OPS is provided by a number of components:

- TMS security includes the Virtual Private Network, used on the links between the centre and the post office outlets, to provide confidentiality and authentication over the network. For further details refer to *TMS Architecture Specification.*
- Within the OPS environment, specific directories on the hard discs are encrypted, but any LAN communications within the outlet are in clear. For further details refer to *OPS Architecture Specification.*
- Riposte provides Cyclic Redundancy Checking on all messages within the TMS environment. For further details, refer to *TMS Architecture Specification.*
- Cryptography provides a higher level of security for messages within the TMS environment. It is this element of security that is covered in this appendix.

  Cryptography is used in several parts of the Horizon system to provide security services. For example, a Virtual Private Network is implemented between the Data Centres and the post office outlets (gateway) to provide confidentiality, authentication and integrity over the network. It should be noted that LAN communications within an outlet are in clear.

  The details required to develop a new application that uses key distribution and cryptographic functions are highly specific to the individual business requirement. Each application using cryptography operates within a specific cryptographic context. The underlying ICL Pathway Key Management Service (KMS) supports these applications by managing the generation, delivery and life cycle of all cryptographic key material. KMS provides a context-specific

FUJ00001379

Cryptographic Functions API to applications. These interfaces rely on the underlying key management service and cannot be used in isolation.

It is important to understand the context into which a new application is to be introduced; this is illustrated in Figure B-1.

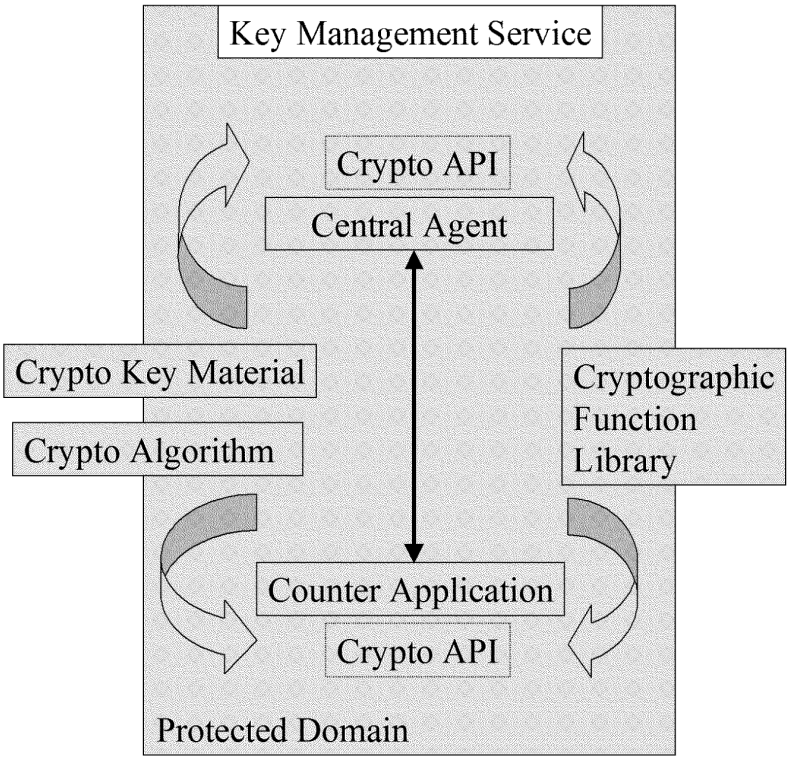

Figure B-1 Application Cryptographic Context

Each application exists in a Protection Domain, which is managed by KMS. Cryptographic key material is distributed by KMS, which also provides the cryptographic functions needed by any application that invokes encryption and decryption.

Section B.3 describes the key management domains in use in the Horizon system, and how Protection Domains are managed. The distribution of the keys involved is described in section B.4 with the environment for key management centrally and at the counter outlined in section B.5. This section also identifies the role played by Riposte in the key management process.

If an application requires a new cryptographic context, for example using a new algorithm, the impact on KMS and these existing domains has to be evaluated. The information required for such an evaluation is given in section B.6. Section B.7 contains the definition of the Crypto API interfaces.

Section B.8 describes the use of Windows NT Event Management interfaces that are used by KMS and the application to record events that occur during

FUJ00001379

use of the cryptographic functions. Section B.9 describes the way in which an application calls the Cryptographic Functions API.

There are some restrictions that need to be considered in planning the usage of the cryptographic API and these are listed in section B.10.

B.3 Key Management Domains

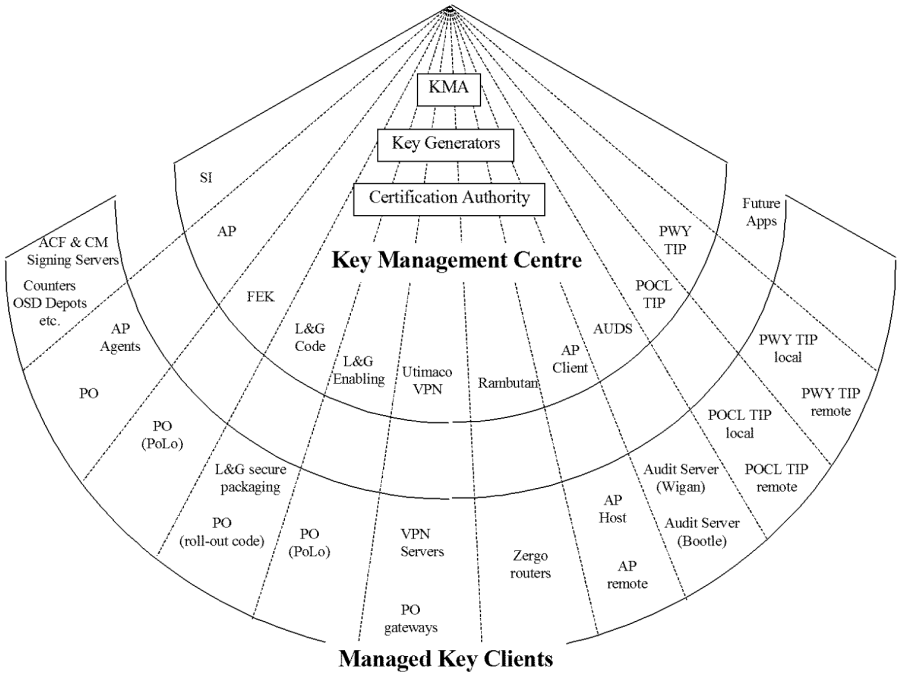

Figure B-2 Key Management domains

Figure B-2 shows how key management emanates from a single point of control, fanning out along segments that correspond to the various uses of cryptography to the many points at which keys are used. Each Managed Key Client operates in its own cryptographic context known as a *Protection Domain*, managed from the Key Management Centre. For example, TIP cryptographic applications are considered under the protection domains POCL TIP and PWY TIP, one corresponding to authentication of POCL to Pathway and the other corresponding to the authentication of Pathway to POCL.

There are two major divisions, shown in Figure B-2 as horizontal sections. They are as follows:

- The Key Management Centre domain encompasses the apparatus that the ICL Pathway Security Manager uses to control the use of keys.
- The Managed Key Clients domain encompasses all platforms on which managed keys are used for cryptography. These include PO Outlets, campus and remote gateway platforms.

FUJ00001379

The Key Management Service supports each protection domain by supplying the appropriate key material required by each Key Management Client.

The protection domains are shown as radial segments in Figure B-2. Each protection domain represents the 'space' in which keys are managed for a particular cryptographic application; for example, 'AP' is the protection domain for Automated Payment keys. Each protection domain may be identified with a specific purpose. Several protection domains may use the same type of key, but the key value is unique. The key value in a protection domain will change each time the key is replaced. The domains currently in use support ICL Pathway's distribution process, the Post Masters' secure logon needed to unlock message encryption, Automated Payment and the particular requirements of cards such as those provided by L&G, as well as the security used in communications and audit. The security of the TIP interface both within ICL Pathway and to the POCL domain is also covered.

KMS manages the complete key life cycle from generation, through activation to deactivation and destruction. There is a user interface for the ICL Pathway Key Manager to force an early key change and where necessary revoke a key. When a key is changed, the keys may remain in an active state for some time until it is known that the old keys are no longer required. It is possible therefore for more than one key set to be in use at a time, particularly during a transition period. In this case, both the current key set and the previous key set exist at the same time, and KMS controls which key set is to be used. The context indicates which key set is to be used and the application does not need to deal with the implications of key changes or key selection (see section B.7.2).

B.4 Key Distribution

For applications within OPS/TMS, all confidential key material is distributed in two parts using two logically discrete routes: Riposte and a second delivery channel. In the case of the AP protection domain, for example, the PostMaster's Memory Card (PMMC) provides one part of the key material and Riposte the other part. Key material to be distributed by Riposte is generated and held in a Key Management Application database that is architecturally equivalent to a Host. Key material is distributed by a set of interactive Key Management Agents, which listen for a request to deliver material and then extract new key material from this database and transmit it via Riposte across the Virtual Private Network (VPN) to the relevant platform. The process is summarised in Figure B-3.

FUJ00001379

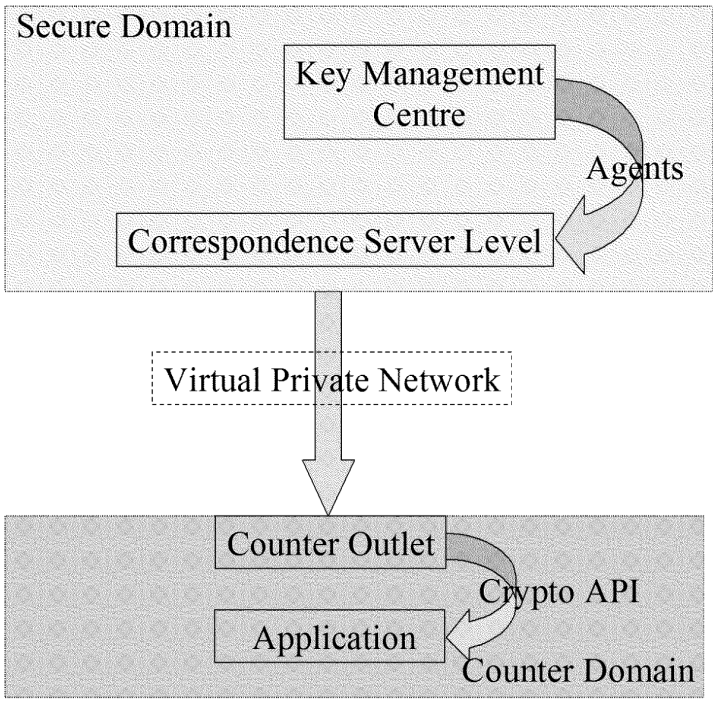

Figure B-3 Key generation and distribution

B.5 Key Management Client Environment

Each platform that is involved in encryption or decryption operations has a local Riposte Client service running on it. Key Management software running on the Client platform obtain and act on new key material and on the appropriate messages that arrive in the Riposte Message Store, for example by unloading a previous signing key and loading its replacement. The Cryptographic Functions API is provided to enable the applications to perform the required cryptographic operations using the current key material appropriate to their specified cryptographic context.

FUJ00001379

B.6 New Cryptographic Context

If a new application requires a new cryptographic context and associated Key Management Protection Domain, details must be supplied to ICL Pathway to enable the scoping of the new KMS Protection Domain and to define the configuration data required to support it.

The necessary details include:

- Cryptography algorithms to be used
- Types and volumes of key material
- How the keys are to be generated
- If appropriate, the Certification Authority for the keys
- The type(s) of Client platform
- The specific cryptographic material required on each platform
- Performance features of the application

Even if the requirements of a new application match existing facilities, in addition to setting up configuration data, ICL Pathway must update the Key Management Service for a new protection domain and perform testing before the application can be released to the live estate. The method for providing these facilities would be agreed as part of the *Security Evaluation Review* process. See Appendix C, *System Management,* for further information.

The algorithms supported to date for existing applications are shown in the table below:

Algorithm	*Purpose*
Red Pike	Encryption using symmetric 64 bit key
DSA	Sign/verify using 768 bit key
SHA	Secure Hash Algorithm

Other algorithms are used by KMS, but they are only supported on customised interfaces to and from specific third party applications, and as such, are not included as part of the generalised Crypto API. Any application-specific requirements are identified as necessary; for example, for the L&G domain, KMS identifies the appropriate key for the relevant software.

Further algorithms or key generation capability can be considered during the *Security Evaluation Review* process for bespoke implementation, or as enhancements to the Crypto API itself. See Appendix C, *System Management,* for further information.

FUJ00001379

B.7 Cryptographic Function Library

A dynamic link library is provided that contains the Cryptographic Functions API. This allows an application to:

- Encrypt data
- Decrypt data
- Add a digital signature to a message
- Verify a message's digital signature

These functions allow an application to encrypt all or part of a transient message and add a digital signature to it before it is committed to the message store and replicated to the centre. An agent would then verify that the digital signature was correct before decrypting the message. The reverse is of course true. In both cases the key involved would have been distributed already by KMS via Riposte.

The processes are summarised in Figure B-4.

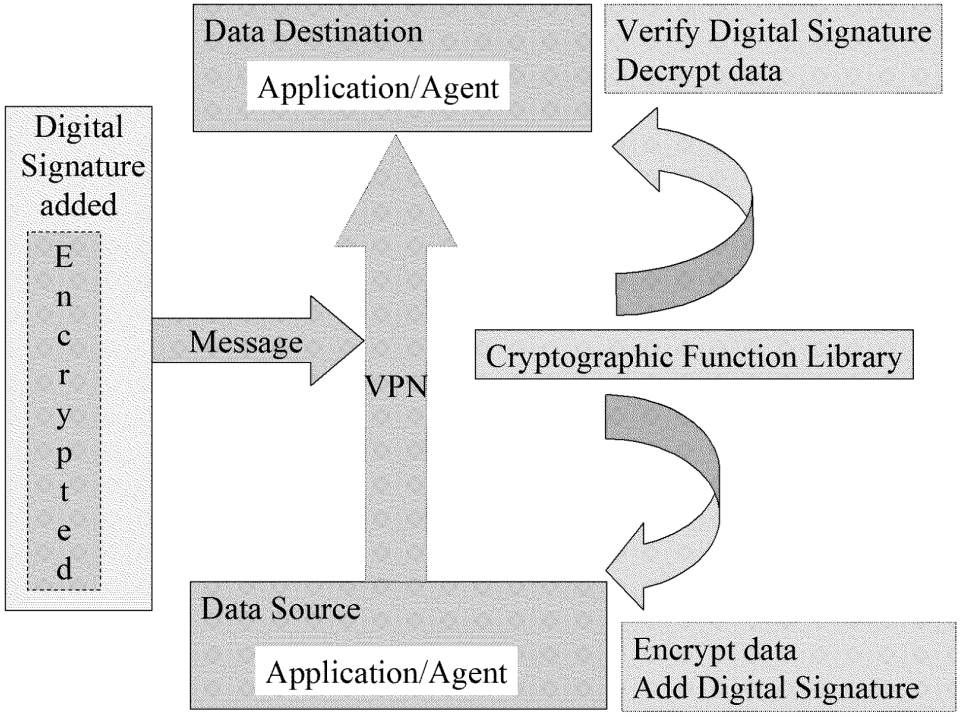

Figure B-4 Cryptographic functions

FUJ00001379

B.7.1 Application context

Each application or agent using the Cryptographic Functions API is required to specify the context within which it is operating each time a function is used. The context includes a protection domain and the name of the key owner. In the case of signing, encryption or data decryption this enables the key to be used to be determined. In the case of verification or file decryption it enables a check to be performed that the key supplied with the data was the correct key for the context.

A single context can support either encryption or sign/verify. If an application requires both, it must use two contexts.

B.7.2 Multiple Keys

For most cryptographic relationships a single key set is in use at any one time. For some cryptographic relationships it is possible to use more than one key set at a time: the current key set or the previous key set. The context indicates which key set is to be used and the application does not need to deal with the implications of key changes or key selection.

The issue of key changes and refresh cycles will be covered during the *Security Evaluation Review* process. See Appendix C, *System Management,* for further information.

FUJ00001379

B.7.3 Cryptographic functions

Table B-1 contains a list of the available functions. The availability of specific functions depends on the context (CRY_CONTEXT). For example, the encryption interfaces are not available within a context that supports signing, and vice versa.

Function	*Description and comments*	*Parameters*	*Return values*
crySignStart	Initialise signing for a context. Loads the primary key for the specified context.	context Context of signer.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key does not exist. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
crySignData	Sign data. The content of the signature block must be transmitted from the signing to the verifying application where it is presented as an input parameter to the **cryVerifyData** function. The binary signature block may be converted to and from a format suitable for transmission via Riposte using the functions **cryBinToB64** and **cryB64ToBin**.	context Context of signer. lData Length of data to be signed. pData Pointer to buffer holding data to be signed. lSignatureBlock Length of generated signature block. pSignatureBlock Pointer to generated signature block.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_BUFFER_TOO_SMALL Buffer too small (to hold signature block). CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key does not exist. CRY_PKC_NOT_FOUND Public key certificate does not exist. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
crySignFile	Sign file(s). The output is created if it does not	context Context of signer.	**Success:** CRY_OK

FUJ00001379

	exist. A signature block is written to the output file. If inlineData is set to TRUE then data from the file being signed is included in the output file. The content of the output file must be transmitted from the signing to the verifying application where it is presented as an input parameter to the **cryVerifyFile** function	dataPath Pathname(s) of file(s) to be signed. offset Offset within file at which processing is to start. len Number of bytes within file to be processed. inlineData Indicates whether or not data is to be included in output file. destPath Pathname of output file.	**Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key does not exist. CRY_PKC_NOT_FOUND Public key certificate does not exist. CRY_FILE_NOT_FOUND File does not exist. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
crySignStop	Complete signing for a context.	context Context of signer.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryVerifyStart	Initialise verification for a context. Loads the primary key for the specified context if context indicates symmetric algorithm.	context Context of signer.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryVerifyData	Verify data. The content of the signature block must have been generated by a call to the **crySignData** function (see above) and transmitted from the	context Context of signer. lData Length of data to be verified. pData Pointer to buffer	**Success:** CRY_OK Information: CRY_INVALID_SIGNATURE Signature not generated by key in signature block. CRY_KEY_REVOKED

FUJ00001379

	signing application.	holding data to be verified. lSignatureBlock Length of signature block. pSignatureBlock Pointer to signature block. revocationReason Reason why key has been revoked.	Key has been revoked. **Failure:** CRY_WRONG_KEY_USED Signature not generated by a current key for this context. CRY_INVALID_CONTEXT Invalid context. CRY_INVALID_SIGNATURE_BLOCK Invalid signature block. CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key does not exist. CRY_PKC_EXPIRED Public key certificate has expired. CRY_PKC_FOUND Public key not found. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryVerifyFile	Verify file(s). Where more than one filename is specified the order is significant and must be the same as that used when the signature was generated.	context Context of signer. dataPath Pathname(s) of file(s) to be verified. offset Offset within file at which processing is to start. len Number of bytes within file to be processed.	**Success:** CRY_OK Information: CRY_INVALID_SIGNATURE Signature not generated by key in signature block. CRY_KEY_REVOKED Key (specified in signature block) has been revoked. **Failure:** CRY_WRONG_KEY_USED Signature not generated by a current key for this context.

FUJ00001379

		inlineData Indicates whether or not data is included in input file. sourcePath Pathname of input file. revocationReason Reason why key has been revoked	CRY_INVALID_CONTEXT Invalid context. CRY_INVALID_SIGNATURE_BLOCK Invalid signature block. CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key (specified in signature block) does not exist. CRY_FILE_NOT_FOUND File does not exist. CRY_PKC_EXPIRED Public key certificate has expired. CRY_PKC_FOUND Public key not found. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryVerifyStop	Complete verification for a context.	context Context of signer.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryEncryptStart	Initialise encryption for a context. Loads the primary key for the specified context if context indicates symmetric algorithm.	context Context of signer.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_NO_RESOURCE Resource not available.

FUJ00001379

			CRY_KEY_NOT_FOUND Key does not exist. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryEncryptData	Encrypt data. The encrypted data is returned in the buffer pointed to by pEncryptedData. The encrypted data may be decrypted using the **cryDecryptData** function.	context Context of encryptor. lData Length of (decrypted) data. pData Pointer to buffer holding data to be encrypted. lEncryptedData Length of encrypted data. pEncryptedData Pointer to encrypted data.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_BUFFER_TOO_SMALL Buffer too small (to hold encrypted data). CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key does not exist. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryEncryptFile	Encrypt file. The file destPath is created if it does not exist. The content of the file destPath must be transmitted from the encrypting to the decrypting application where it is presented as an input parameter to the **cryDecryptFile** function.	context Context of encryptor. sourcePath Pathname of file to be encrypted. offset Offset within file at which processing is to start. len Number of bytes within file to be processed. destPath Pathname of file to contain encrypted source file.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key does not exist. CRY_FILE_NOT_FOUND File does not exist. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryEncryptStop	Complete encryption for a context.	context Context of encryptor.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context.

FUJ00001379

			CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryDecryptStart	Initialise decryption for a context. Loads the primary key for the specified context if context indicates symmetric algorithm.	context Context of signer.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key does not exist. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryDecryptData	Decrypt data. The decrypted data is returned in the buffer pointed to by pData.	context Context of signer. lEncryptedData Length of encrypted data. pEncryptedData Pointer to encrypted data. lData Length of (decrypted) data. pData Pointer to buffer holding (decrypted) data.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_BUFFER_TOO_SMALL Buffer too small (to hold data). CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND Key does not exist. CRY_INVALID_ENCRYPTION_BLOCK Encryption block is not valid. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryDecryptFile	Decrypt file. The content of the (section of) the file specified by sourcePath that is to be decrypted must have been generated by a call to the **cryEncryptFile** function (see above)	context Context of encryptor. sourcePath Pathname of file to be decrypted. offset Offset within file at which processing is	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_NO_RESOURCE Resource not available. CRY_KEY_NOT_FOUND

FUJ00001379

	and transmitted from the encrypting application.	to start. len Number of bytes within file to be processed. destPath Pathname of file to contain decrypted source file.	Key does not exist. CRY_WRONG_KEY_USED Key not valid in this context. CRY_FILE_NOT_FOUND File does not exist. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryDecryptStop	Complete decryption for a context.	context Context of encryptor.	**Success:** CRY_OK **Failure:** CRY_INVALID_CONTEXT Invalid context. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter.
cryBinTo64	Convert binary data to base-64 encoding. The converted data, terminated by a null character, is returned in the buffer pointed to by pOut.	lIn Length of binary data to be converted. pIn Pointer to binary data. lOut Length of converted data. pOut Pointer to buffer to receive converted data.	**Success:** CRY_OK **Failure:** CRY_BUFFER_TOO_SMALL Buffer too small (to hold converted data). CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter.
cry64ToBin	Convert base-64 encoded data to binary. The converted data is returned in the buffer pointed to by pOut.	pIn Pointer to base-64 encoded data. lOut Length of converted data. pOut Pointer to buffer to receive converted	**Success:** CRY_OK **Failure:** CRY_BUFFER_TOO_SMALL Buffer too small (to hold converted data). CRY_INVALID_DATA Data is invalid. CRY_MEMORY_ERROR

FUJ00001379

		data.	Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.
cryHashData	Generates hash value for a variable length buffer. A hash of the data is returned in the buffer pointed to by pOut.	lIn Length of data. pIn Pointer to data._ lOut Length of data. pOut Pointer to buffer.	Success: CRY_OK Failure: CRY_NO_RESOURCE Resource not available. CRY_INVALID_PARAMETER Invalid parameter. CRY_BUFFER_TOO_SMALL Buffer too small. CRY_MEMORY_ERROR Unable to obtain memory. CRY_INVALID_PARAMETER Invalid parameter. CRY_ERR_OTHER Other error.

Table B-1 Cryptographic functions - continued

B.7.4 Return values

Each function returns a value to the caller. The value indicates success, provides information or indicates failure:

- A single success value (CRY_OK) is defined for all functions and indicates that the function has completed its desired action and is returning success to the caller.
- Zero or more information values are defined for a function. These indicate that the function has completed its desired action and is returning other than success to the caller.
- One or more failure values are defined for a function and indicate that the function has been unable to complete its desired action.

Information and failure codes are listed in the following tables.

FUJ00001379

B.7.5 Information codes

Table B-2 gives the values that may be returned for information.

Symbol	*Message*	*Description*
CRY_INVALID_SIGNATURE	Signature not valid.	The signature was not generated by the key identified in the signature block. The data is either corrupt or from an unauthorised source.
CRY_KEY_REVOKED	Key has been revoked.	The key identified in the signature block has been revoked. The data has either been long-delayed in transit or it comes from an unauthorised source.
CRY_PKC_EXPIRED	Public Key Certificate has expired	The Public Key Certificate for the key identified in the signature block has expired. The data has either been long-delayed in transit or it comes from an unauthorised source.

Table B-2 Status codes: information

It is the responsibility of the application to take the appropriate action when these conditions are encountered. This includes identifying the appropriate follow up actions; for example, alerting the ICL Pathway Security Manager that a security relevant event has occurred. The Cryptographic functions are not able to recognise whether these conditions are security relevant and so do not log NT events for these conditions.

B.7.6 Failure Codes

Table B-3 illustrates the sorts of values that may be returned on failure.

The codes returned on failure may need to be extended if new applications are introduced and this will be assessed as part of the *Security Evaluation Review* process. See Appendix C, *System Management*, for further information.

The errors that are dealt with by the cryptographic functions themselves are identified in Table B-4.

Symbol	*Message*	*Description*
CRY_BUFFER_TOO_SMALL	Buffer too small.	The buffer size specified is too small for the data requested. The location specifying the buffer size has been updated to give the minimum buffer size required.
CRY_ERR_OTHER	Other failure.	An unanticipated error condition has occurred. A diagnostic message giving details of this will have been written to the event log. This is a 'catch all' error condition that deals with errors than are not explicitly identified otherwise.
CRY_FILE_NOT_FOUND	File not found.	A file used as input to the cryptographic

FUJ00001379

		operation cannot be located.
CRY_INVALID PARAMETER	Invalid Parameter	One of the parameters supplied by the caller is incorrect.
CRY_INVALID_CONTEXT	Invalid context.	The context specified by the caller has not been recognised, or is not available on this platform.
CRY_INVALID_DATA	Invalid data	The data supplied as a parameter is invalid, for example not base-64 encoded data.
CRY_INVALID_ENCRYPTION_BLOCK	Invalid encryption block.	The encryption block cannot be analysed. The data is either corrupt or from an unauthorised source.
CRY_INVALID_SIGNATURE_BLOCK	Invalid signature block.	The signature block cannot be analysed. The data is either corrupt or from an unauthorised source.
CRY_KEY_NOT_FOUND	Key does not exist.	A key that is required to perform the cryptographic operation cannot be located.
CRY_MEMORY_ERROR	Unable to obtain memory	Failure to obtain memory to perform the operation.
CRY_NO_RESOURCE	Resource not available.	A resource that supports the loading or usage of keys is temporarily unavailable. The caller is expected to retry periodically.
CRY_PKC_NOT_FOUND	Public key certificate does not exist.	A public key certificate that is required to perform the cryptographic operation cannot be located.
CRY_WRONG_KEY_USED	Wrong key used.	The signature was not generated using a key owned by the caller-specified context.

Table B-3 Status codes: failure

B.8 Event Logging

The cryptographic functions log messages relating to failures to the NT event log. Each event generated is classified according to its severity and security relevance. This classification of events determines which events are reported to the data centre, and the routing of any alerts between operational support and security management. The specification of the event filters is under the control of the ICL Pathway Security Manager.

FUJ00001379

Note that not all failure codes result in the generation of NT events, for example, CRY_NO_RESOURCE and CRY_BUFFER_TOO_SMALL can be used by applications to control further actions. It is also the responsibility of the applications to log their own security relevant events. In particular, the applications are responsible for acting on the information response code CRY_INVALID_SIGNATURE in the same way that the application is responsible for actually calling the cryptography API in the first place. Application writers must also take into account the frequency of potential failure events and decide on the appropriate logging policy.

For performance reasons it is not appropriate to swamp the NT event log with every instance of a failure; for example, it is not desirable to record every event during a bulk load or harvest of data. This issue can be raised as part of the *Performance Review*. See Appendix C, *System Management,* for further information.

Table B-4 gives an example set of NT events generated by KMS.

A similar set of NT events would need to be developed for each application and discussed with the ICL Pathway Security Manager as part of the *Security Evaluation Review* process. See Appendix C, *System Management,* for further information.

Symbol	*Event Message*	*Support Notes*
CRY_INVALID_CONTEXT	Invalid context specified.	Check context configuration.
CRY_KEY_NOT_FOUND	Key for context %1 does not exist.	Check key for specified context available.
CRY_PKC_NOT_FOUND	Public key certificate for context %1 does not exist.	Check public key certificates for specified context available.
CRY_INVALID_SIGNATURE_BLOCK	Invalid signature block specified.	Data may have been received from an unauthorised source.
CRY_WRONG_KEY_USED	Signature not generated by a current key for context %1.	Object signed using wrong key. Check keys available to signer in specified context. Data may have been received from an unauthorised source.
CRY_FILE_NOT_FOUND	File %1 does not exist.	File or directory may have been accidentally deleted.
CRY_ERR_OTHER	Called function %1 returned error code %2.	This is returned as a failure event to the application. The NT event log contains further details

FUJ00001379

		which are forwarded to ICL Pathway System Management and Security Manager as appropriate. See Appendix C, *System Management, for details.*

Table B-4 Event logging: failure

The event codes that are not considered by the Cryptographic Functions as failures are given in section B.7.6. It is the responsibility of the application to resolve these errors.

FUJ00001379

B.9 Calling the functions

The Cryptographic functions that applications or agents may use can be called from C and from Visual Basic. They are provided in a dynamic link library. Header files (**cry.h** and **cry.bas**) are provided for these languages which contain:

- prototypes for each of the functions
- definitions of types used for function parameters
- defines for constants used in function parameters and return values

There is a strict order in which the functions must be called by an application. For example:

1. Start function: **crySignStart**, is called before performing signing operations. This performs any initialisation required, such as loading keys and checking that the algorithm modules required are available.
2. Action function(s): **crySignData**
3. Termination function: a stop function, **crySignStop**, is called to perform any termination actions required.

B.10 Restrictions

There are some restrictions that need to be considered in planning the usage of the cryptographic API:

- The cryptographic API is only available on platforms running Windows NT
- The cryptographic API relies on the underlying Key Management Service and cannot be used in isolation.
- There needs to be a local Riposte Message Service running on each platform using the cryptographic API, but it is not dependent on the Riposte Desktop.
- The cryptographic API can be used by multiple processes running concurrently on the same platform, but it is the responsibility of the calling application to call currently on a single thread within a process. The API is not thread safe.
- Application writers should produce code that is as defensive as possible against extensions to the list of information and failure codes. For example, the list may need to be extended to support new applications that are introduced.