

Key Management Interoperability Protocol Usage Guide Version 1.0

Committee Draft 0905 / Public Review 0201

18 March 2010

5 November 2009

Specification URIs:	
http://docs.oasis-open.org/kmip/ug/v1.0/cd09cd05/kmip-ug-1.0-cd-0905.html	Field Code Changed
http://docs.oasis-open.org/kmip/ug/v1.0/cd09cd05/kmip-ug-1.0-cd-0905.doc (Authoritative)	Field Code Changed
http://docs.oasis-open.org/kmip/ug/v1.0/ <u>cd09</u> cd05/kmip-ug-1.0-cd- <u>09</u> 05.pdf	Field Code Changed
Previous Version:	Tield Code Changed
http://docs.oasis-open.org/kmip/ug/v1.0/cd05/kmip-ug-1.0-cd-05.html	
http://docs.oasis-open.org/kmip/ug/v1.0/cd05/kmip-ug-1.0-cd-05.0dc	
N/A	
Latest Version:	
http://docs.oasis-open.org/kmip/ug/v1.0/kmip-ug-1.0.html	
http://docs.oasis-open.org/kmip/ug/v1.0/kmip-ug-1.0.doc	
http://docs.oasis-open.org/kmip/ug/v1.0/kmip-ug-1.0.pdf	
Technical Committee:	
OASIS Key Management Interoperability Protocol (KMIP) TC	
Chair(s):	
Robert Griffin, EMC Corporation <robert.griffin@rsa.com> Subbash Sankuratrinati. NetAnn <subbash sankuratrinati@netann.com=""></subbash></robert.griffin@rsa.com>	
Ealtor(s):	
Polated work:	
This specification replaces or supersedes:	
None	
This specification is related to:	
Key Management Interoperability Protocol Specification Version 1.0	
Key Management Interoperability Protocol Lise Cases Version 1.0	
Deplayed VML Nemocrosolo):	
None	
Abstract:	
This document is intended to complement the Key Management Interoperability Protocol	
Specification by providing guidance on now to implement the Key Management Interoperability Protocol (KMIP) most effectively to ensure interoperability	
kmip-ug-1.0-cd-09 18 March 201005-	5 November 2009
Copyright © OASIS® 2010,2009. All Rights Reserved. Page 1 of 46	

Style Definition: Heading 3,H3,h3,Level 3Topic Heading: Indent: Left: 0 pt, Hanging:43.2 pt, Tab stops: 0 pt, List tab + Not at 36 pt

Style Definition: Heading 4,H4,h4,First Subheading: Indent: Left: 0 pt, Hanging: 36 pt, Tab stops: Not at 43.2 pt

Status:

This document was last revised or approved by the Key Management Interoperability Protocol TC on the above date. The level of approval is also listed above. Check the "Latest Version" or "Latest Approved Version" location noted above for possible later revisions of this document.

Technical Committee members should send comments on this specification to the Technical Committee's email list. Others should send comments to the Technical Committee by using the "Send A Comment" button on the Technical Committee's web page at http://www.oasis-open.org/committees/kmip/.

For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the Technical Committee web page (http://www.oasis-open.org/committees/kmip/ipr.php.)

The non-normative errata page for this specification is located at http://www.oasisopen.org/committees/kmip/.

Notices

Copyright © OASIS® 20102009. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full Policy may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Committee Specification or OASIS Standard, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification.

OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this specification by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this specification. OASIS may include such claims on its website, but disclaims any obligation to do so.

OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Committee Specification or OASIS Standard, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.

The names "OASIS", "KMIP" are trademarks of OASIS, the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see http://www.oasis-open.org/who/trademark.php for above guidance.

Table of Contents

1Inti	roduction		Formatted: Don't adjust space betwee	en Latin
1.1	Terminology	7	and Asian text, Tab stops: 24 pt, Left	
1.2	Normative References	7	Field Code Changed	[1]
1.3	Non-normative References	10	Field Code Changed	[[2]
2 Ass	sumptions		Field Code Changed	[[3]
2.1	Island of Trust		Field Code Changed	[[4]
2.2	Message Security		Field Code Changed	[5]
2.3	State-less Server		Field Code Changed	[6]
2.4	Extensible Protocol		Field Code Changed	[17]
2.5	Server Policy		Field Code Changed	
	Support for Cryptographic Objects	11	Field Code Changed	
<u>2.6</u>	Support for Cryptographic Objects	<u></u> 2.6	Field Code Changed	<u> [9]</u>
	Client-Server Message based Model			[[10]
<u>2.7</u>	Client-Server Message-based Model	<u>2.7</u>	Field Code Changed	([11]
			Field Code Changed	[12]
<u>2.8</u>	Synchronous and Asynchronous Messages		Field Code Changed	[13]
	-Support for Intelligent Clients and "Key Using Devices"			
2.9	Support for "Intelligent Clients" and "Key Using Devices"	<u>2.9</u> 12	Field Code Changed	[14]
0.40	Batched Requests and Responses			
2.10	Batched Requests and Responses	<u></u>	Field Code Changed	([15]
2 11	Reliable Message Delivery	2 11		
2.11			Field Code Changed	([16]
2 12	Large Responses	212	Field Code Changed	[17]
<u></u>	Key Life-cycle and Key State		Formatted: TOC 2 toc2 Don't adjust	space
2.13	Key Life-cycle and Key State		between Latin and Asian text, Tab stop	os: 48 pt,
A	Usage Guidelines		Left + Not at 24 pt	
<u>3 Usa</u>	age Guidelines		Field Code Changed	[18]
Aut	hentication		Formatted: TOC 1,toc1, Don't adjust	space
<u>3.1</u>	Authentication	<u></u>	between Latin and Asian text, Tab stop Left + Not at 48 pt)s: 24 pt,
	-Authorization for Revoke, Recover, Destroy and Archive Operations		Eield Code Changed	([40]
<u>3.1</u>	.1 Credential	<u></u>	Field Code Changed	[[19]
				[[20]
3.2	Authorization for Revoke, Recover, Destroy and Archive Operations		between Latin and Asian text. Tab stop	space os: 60 pt.
	Using Natify and Dut Or antiger		Left + Not at 48 pt	
3.3	Using Notify and Put Operations	<u>3.5</u>	Field Code Changed	[[21]
2.4			Field Code Changed	[[22]
<u>.3.4</u>	- Template	3.0	Field Code Changed	
35	Key State and Times	3.7	Field Code Changed	<u>[[23]</u>
0.0	Archive Operations			[24]
3.6	Template	3.8	Field Code Changed	[[25]
A	Message Extensions	16	Field Code Changed	([26]
3.6	.1 Template Usage Examples		Formatted: TOC 3,toc3, Don't adjust	space
	Unique Identifiers	17	Left + Not at 48 pt	.s. ov pt,
3.7	Archive Operations	<u></u> 3.10	Field Code Changed	[[27]
	Result Message Text		Field Code Changed	
kmip-ua-1 (0-cd- <mark>09</mark>	18 March 201005	5 November 2009	([28]
Copyright @	© OASIS® 2010,2009. All Rights Reserved.	Page 4 of 46		

	Message Extensions	<u></u> 3.11		Field Code Changed	
	-Query		/		(
.9	Unique Identifiers	3.12	7	Field Code Changed	
	Canceling Asynchronous Operations		/		(
.10	Result Message Text	3.13	7	Field Code Changed	
	Multi-instance Hash	18	1		(
.11	Query	<u>3.14</u>	- 2	Field Code Changed	
	Returning Related Objects	18	/	Tield Code Changed	L
12	Canceling Asynchronous Operations	3.15		Field Code Changed	
	Reducing Multiple Requests through the Use of Batch		/	Field Code Changed	(
13	Multi-instance Hash	3 16		Field Code Channed	
. 15	Maximum Message Size	19	/	Field Code Changed	(
14	Peturning Delated Objects	2 47			
. 14	Returning Related Objects.	<u>ə. 17</u> 10	/	Field Code Changed	(
			/		
.15	Reducing inuitiple Requests through the Use of Batch	<u></u> న.18	7	Field Code Changed	
	-Locale Queries				
.16	Maximum Message Size.	<u></u> 3.19	7	Field Code Changed	(
	ID Placeholder	19			
.17	Using Offset in Re-key and Re-certify Operations	<u></u> 3.20	7	Field Code Changed	<u> </u>
	Key Block	<u> </u>			<u></u>
.18	Locate Queries	3.21	7	Field Code Changed	
	Using Wrapped Keys with KMIP	20	1		<u> </u>
19	ID Placeholder	3 21 14		Formatted: TOC 2 toc2 Don't	adjust sna
	-Encrypt-only Example with a Symmetric Key as an Encryption Key for a Get Request a	nd	×	between Latin and Asian text, Ta	ab stops:
espo	nse	21	\sim	Left + Not at 72 pt	·
20	21.2 Encrypt-only Example with a Symmetric Ke	ev Block	<u>``</u>	Field Code Changed	
	as an Encryption Key for a Register Request and Response	22	>	Field Code Changed	
21	Light Wrappod Kove with KMIP	2 21 2			
.21	Encrypt only Example with an Asymmetric Key as an Encryption Key for a Cat Request		7	Field Code Changed	
) oeno	nee	23			
2 D	1.1 Enorupt4 MAC only Example with a Symmetrican HMAC K	20			
J.Z	I. <u>Enclypta</u>	y as an	7	Field Code Changed	
			/		
3.2	<u>1.2 Encrypt-only Example with a Symmetric Key as an Encryption Key for a Register</u>		7	Field Code Changed	
Rec	uest and Response	<u></u> 3.21.5			
	Registering a wrapped key as an Opaque Oryptographic Object	24	/		
3.2	1.3 Encrypt-only Example with an Asymmetric Key as an Encryption Key for a Get Re	<u>quest</u>		Formatted: TOC 3,toc3, Don't a	adjust spa
and	Response	<u></u> 3.22	\mathbf{x}	between Latin and Asian text, 1a	ab stops:
	Object Group	24	<u>``</u>		
3.2	1.4 MAC-only Example with an HMAC Key as an Authentication Key for a Get Reque	st and		Field Code Changed	
Res	sponse <u>3.23</u> Certify and R	e-certify	>	Field Code Changed	(.
	25				
<u>3.2</u>	1.5 Registering a Wrapped Key as an Opaque Cryptographic Object	<u></u> 3.24	1	Field Code Changed	(
	———Specifying Attributes during a Create Key Pair Operation	25	11	Formatted: TOC 2,toc2, Don't a	adjust spa
.22	Object Group	<u>3.24.1</u> •	1	between Latin and Asian text, 1a	ab stops:
	-Example of Specifying Attributes during the Create Key Pair Operation	26			
23	Certify and Re-certify	3 25		Field Code Changed	
	–Registering a Key Pair	26	>	Field Code Changed	<u> </u>
24	Specifying Attributes during a Create Key Pair Operation	3 26	- 1	Field Code Changed	
.24	Specifying Autoutes during a Create Ney Fall Operation	ວ.20 ວຣ	/		<u> </u>
		<u>~~</u> 20	11	Formatted: TOC 3,toc3, Don't a	adjust spa ah stons:
0.0	A A DECEMBER OF OUR OF DECEMBER AND A SAMPLE AND A DECEMBER	7		NOT THE REAL PROPERTY AND A STATE OF A STATE	in stops.
3.24	4.1 Example of Specifying Attributes during the Create Key Pair Operation	<u></u> 3.2/•		Left + Not at 48 pt	
3.24	Example of Specifying Attributes during the Create Key Pair Operation Asymmetric Concepts with Symmetric Keys	<u></u> 27 27		Left + Not at 48 pt	

3.25 Registering a Key Pair	<u></u>	Field Code Changed
Application Specific Information		Field Code Changed
3.26 Non-Cryptographic Objects	<u></u>	Field Code Changed
Mutating Attributes	28	Field Code Changed
3.27 Asymmetric Concepts with Symmetric Keys		Field Code Changed
2.00 Application Creatile Information	2 20 4	
3.28 Application Specific Information	<u></u>	Field Code Changed
2.20 Mutating Attributes	2 21	Formatted: IOC 2,toc2, Don't adjust space
5.29 Mutating Attributes	31	Left + Not at 72 pt
3.30 Interoperable Key Naming for Tape	3.32	Field Code Changed
Certificate Renewal, Update, and Re key	31	Field Code Changed
3.30.1 Native Tape Encryption by a KMIP Client		Field Code Changed
Key Encoding		Field Code Changed
3.31 Revocation Reason Codes	<u>3.33.1</u>	
AES Key Encoding		Field Code Changed
3.32 Certificate Renewal, Update, and Re-key	<u></u>	Field Code Changed
Triple-DES Key Encoding		Formatted: TOC 3,toc3, Don't adjust space
3.33 Key Encoding	<u>4</u>	Left + Not at 48 pt
Deferred KMIP Functionality	<u>36</u>	Field Code Changed
3.33.1 AES Key Encoding		Field Code Changed
Implementation Conformance		
3.33.2 Triple-DES Key Encoding		Formatted: TOC 2,toc2, Don't adjust space between Latin and Asian text. Tab stops: 48 pt
Acronyms		Left + Not at 72 pt
3.34 Using the Same Asymmetric Key Pair in Multiple Algorithms		Field Code Changed
Acknowledgements		Field Code Changed
<u>4 Deterred KMIP Functionality</u>	38	Field Code Changed
5 Implementation Conformance		Field Code Changed
	40	
A. Actoriyins	41	Formatted: TOC 2,toc2, Don't adjust space between Latin and Asian text. Tab stops: 48 pt
B. Acknowledgements		Left + Not at 24 pt
C. Revision History	45	Field Code Changed
	unutite unutite	Field Code Changed
Tables	100000	Formatted
Table 1: ID Placeholder Prior to and Resulting from a KMIP Operation		Field Code Changed
Table 2: Cryptographic Usage Masks Pairs		
		Field Code Changed
		Field Code Changed
		Field Code Changed
		Formatted [53]

Field Code Changed Field Code Changed

1 **1 Introduction**

This Key Management Interoperability Protocol Usage Guide is intended to complement the Key
 Management Interoperability Protocol Specification [KMIP-Spec] by providing guidance on how to
 implement the Key Management Interoperability Protocol (KMIP) most effectively to ensure
 interoperability. In particular, it includes the following guidance:

- Clarification of assumptions and requirements that drive or influence the design of KMIP and the
 implementation of KMIP-compliant key management.
- 8 Specific recommendations for implementation of particular KMIP functionality.
- 9 Clarification of mandatory and optional capabilities for conformant implementations.
- Functionality considered for inclusion in KMIP V1.0, but deferred to subsequent versions of the standard.
- A selected set of conformance profiles and authentication suites are defined in the KMIP Profiles
 specification [KMIP-Prof].
- 14 Further assistance for implementing KMIP is provided by the KMIP Use Cases for Proof of Concept
- 15 | Testing document [KMIP-UC] that describes a set of recommended test cases and provides the TTLV
- 16 (Tag/Type/Length/Value) format for the message exchanges defined by those use cases.

17 1.1 Terminology

18 For a list of terminologies refer to [KMIP-Spec].

19 1.2 Normative References

20 21	[FIPS186-3]	Digital Signature Standard (DSS), FIPS PUB 186-3, June 2009, http://csrc.nist.gov/publications/fips/fips186-3/fips_186-3.pdf	
22 23	[FIPS197]	Advanced Encryption Standard (AES), FIPS PUB 197, November 26, 2001, http://csrc.nist.gov/publications/fips/fips/197/fips-197.pdf	
24 25	[FIPS198-1]	The Keyed-Hash Message Authentication Code (HMAC), FIPS PUB 198-1, July 2008, http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf	
26 27	[IEEE1003-1]	IEEE Std 1003.1, Standard for information technology - portable operating system interface (POSIX). Shell and utilities, 2004.	
28 29	[ISO16609]	ISO, Banking Requirements for message authentication using symmetric techniques, ISO 16609, 1991.	
30 31 32	[ISO9797-1]	ISO/IEC, Information technology Security techniques Message Authentication Codes (MACs) Part 1: Mechanisms using a block cipher, ISO/IEC 9797-1, 1999.	
33 34 35	[KMIP-Spec]	OASIS Committee Draft <u>1006</u> , Key Management Interoperability Protocol Specification Version 1.0, <u>March 2010November 2009</u> . <u>http://docs.oasis-open.org/kmip/spec/v1.0/cd10ed06/kmip-spec-1.0-cd-1006.doc</u>	 Formatted: Hyperlin
36 37 38	[KMIP-Prof]	OASIS Committee Draft 0504, Key Management Interoperability Protocol Profiles Version 1.0, March 2010 November 2009. http://docs.oasis- open.org/kmip/profiles/v1.0/cd05cd04/kmip-profiles-1.0-cd-0504.doc	 Field Code Changed
39 40	[PKCS#1]	RSA Laboratories, <i>PKCS #1 v2.1: RSA Cryptography Standard</i> , June 14, 2002, http://www.rsa.com/rsalabs/node.asp?id=2125	
41 42	[PKCS#5]	RSA Laboratories, PKCS #5 v2.1: Password-Based Cryptography Standard, October 5, 2006, http://www.rsa.com/rsalabs/node.asp?id=2127	
43 44	[PKCS#7]	RSA Laboratories, PKCS#7 v1.5: Cryptographic Message Syntax Standard. November 1, 1993, http://www.rsa.com/rsalabs/node.asp?id=2129	

kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved. 18 March 201005 Page 7 of 46 5 November 2009

Hyperlink, Font color: Auto

45 46	[PKCS#8]	RSA Laboratories, PKCS#8 v1.2: Private-Key Information Syntax Standard, November 1, 1993, http://www.rsa.com/rsalabs/node.asp?id=2130
47 48	[PKCS#10]	RSA Laboratories, <i>PKCS #10 v1.7: Certification Request Syntax Standard</i> , May 26, 2000, http://www.rsa.com/rsalabs/node.asp?id=2132
49 50	[RFC1319]	B. Kaliski, <i>The MD2 Message-Digest Algorithm</i> , IETF RFC 1319, Apr 1992, http://www.ietf.org/rfc/rfc1319.txt
51 52	[RFC1320]	R. Rivest, <i>The MD4 Message-Digest Algorithm</i> , IETF RFC 1320, Apr 1992, http://www.ietf.org/rfc/rfc1320.txt
53 54	[RFC1321]	R. Rivest, <i>The MD5 Message-Digest Algorithm</i> , IETF RFC 1321, Apr 1992, http://www.ietf.org/rfc/rfc1321.txt
55 56 57	[RFC1421]	J. Linn, <i>Privacy Enhancement for Internet Electronic Mail: Part I: Message Encryption and Authentication Procedures</i> IETF RFC 1421, Feb 1993, http://www.ietf.org/rfc/rfc1421.txt
58 59 60	[RFC1424]	B. Kaliski, <i>Privacy Enhancement for Internet Electronic Mail: Part IV: Key Certification and Related Services</i> , IETF RFC 1424, February 1993, http://www.ietf.org/rfc/rfc1424.txt
61 62 63	[RFC2104]	H. Krawczyk, M. Bellare, R. Canetti, <i>HMAC: Keyed-Hashing for Message</i> <i>Authentication</i> , IETF RFC 2104 ₄ - Feb <u>1997</u> 1007 , http://www.ietf.org/rfc/rfc2104.txt
64 65	[RFC2119]	S. Bradner, <i>Key words for use in RFCs to Indicate Requirement Levels</i> , http://www.ietf.org/rfc/rfc2119.txt, IETF RFC 2119, March 1997.
66 67	[RFC2898]	B. Kaliski, <i>PKCS</i> #5: Password-Based Cryptography Specification Version 2.0, IETF RFC 2898, Sep 2000, http://www.ietf.org/rfc/rfc2898.txt
68 69	[RFC3394]	J. Schaad, R. Housley, Advanced Encryption Standard (AES) Key Wrap Algorithm, IETF RFC 3394, Sep 2002, http://www.ietf.org/rfc/rfc3394.txt
70 71 72	[RFC3447]	J. Jonsson, B. Kaliski, <i>Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1</i> , IETF RFC 3447 Feb 2003, http://www.ietf.org/rfc/rfc3447.txt
73 74	[RFC3629]	F. Yergeau, <i>UTF-8, a transformation format of ISO 10646</i> , IETF RFC 3629, Nov 2003, http://www.ietf.org/rfc/rfc3629.txt
75 76 77	[RFC3647]	S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, <i>RFC3647: Internet</i> <i>X.509 Public Key Infrastructure Certificate Policy and Certification Practices</i> <i>Framework</i> , November 2003, http://www.ietf.org/rfc/rfc3647.txt
78 79 80	[RFC4210]	C. Adams, S. Farrell, T. Kause and T. Mononen, <i>RFC2510: Internet X.509</i> <i>Public Key Infrastructure Certificate Management Protocol (CMP)</i> , September 2005, http://www.ietf.org/rfc/rfc4210.txt
81 82 83	[RFC4211]	J. Schaad, <i>RFC 4211: Internet X.509 Public Key Infrastructure Certificate</i> <i>Request Message Format (CRMF)</i> , September 2005, http://www.ietf.org/rfc/rfc4211.txt
84 85	[RFC4868]	S. Kelly, S. Frankel, <i>Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec</i> , IETF RFC 4868, May 2007, http://www.ietf.org/rfc/rfc4868.txt
86 87	[RFC4949]	R. Shirey, <i>RFC4949: Internet Security Glossary, Version 2</i> , August 2007, http://www.ietf.org/rfc/rfc4949.txt
88 89	[RFC5272]	J. Schaad and M. Meyers, <i>RFC5272: Certificate Management over CMS (CMC)</i> , June 2008, http://www.ietf.org/rfc/rfc5272.txt
90 91 92	[RFC5280]	D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley and W. Polk, <i>RFC</i> 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, May 2008, http://www.ietf.org/rfc/rfc5280.txt
93 94	[RFC5649]	R. Housley, Advanced Encryption Standard (AES) Key Wrap with Padding Algorithm, IETF RFC 5649, Aug 2009, http://www.ietf.org/rfc/rfc5649.txt
95 96 97	[SP800-38A]	M. Dworkin, Recommendation for Block Cipher Modes of Operation – Methods and Techniques, NIST Special Publication 800-38A, Dec 2001, http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf

kmip-ug-1.0-cd-<u>09</u> Copyright © OASIS® <u>2010,</u>2009. All Rights Reserved.

98 99 100	[SP800-38B]	M. Dworkin, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, NIST Special Publication 800-38B, May 2005, http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf
101 102 103 104	[SP800-38C]	M. Dworkin, Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality, NIST Special Publication 800-38C, May 2004, http://csrc.nist.gov/publications/nistpubs/800-38C/SP800- 38C_updated-July20_2007.pdf
105 106 107	[SP800-38D]	M. Dworkin, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, NIST Special Publication 800-38D, Nov 2007, http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf
108 109 110 111	[SP800-38E]	M. Dworkin, Recommendation for Block Cipher Modes of Operation: The XTS- AES Mode for Confidentiality on Block-Oriented Storage Devices, NIST Special Publication 800-38E, <u>Jan 2010</u> , <u>Aug 2009 (draft)</u> , <u>http://csrc.nist.gov/publications/nistpubsdrafts/800-38E/nistdraft-sp-800-38E.pdf</u>
112 113 114 115 116	[SP800-56A]	E. Barker, D. Johnson, and M. Smid, <i>Recommendation for Pair-Wise Key</i> <i>Establishment Schemes Using Discrete Logarithm Cryptography (Revised)</i> , NIST Special Publication 800-56A, March 2007, <u>http://csrc.nist.gov/publications/nistpubs/800-56A/SP800-56A_Revision1_Mar08-</u> <u>2007.pdf</u>
117	[SP800-56B]	E. Barker, L. Chen, A. Regenscheid, M. Smid, Recommendation for Pair-Wise
118		Key Establishment Schemes Using Integer Factorization Cryptography, NIST
119		Special Publication 800-56B, August 2009,
120		http://csrc.nist.gov/publications/nistpubs/800-56B/sp800-
121		<u>300 Hitp://CSrc.Hist.gov/publications/Histpubs/000-30/VSP000-</u> 56A Rovision1 Mar08 2007 ndf
122	[CD000 E7 1]	E Parker W Parker W Purr W Polk and M Smid Personmandations for Key
123	[3F000-37-1]	Management - Part 1: General /Pavised) NIST Special Publication 800-57 part
125 126		1, March 2007, http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1- revised2_Mar08-2007.pdf
127 128 129	[SP800-67]	W. Barker, <i>Recommendation for the Triple Data Encryption Algorithm (TDEA)</i> <i>Block Cipher</i> , NIST Special Publication 800-67, Version 1.1, Revised 19 May 2008, http://csrc.nist.gov/publications/nistpubs/800-67/SP800-67.pdf
130 131 132	[SP800-108]	L. Chen, <i>Recommendation for Key Derivation Using Pseudorandom Functions (Revised)</i> , NIST Special Publication 800-108, October 2009, http://csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf
133 134 135 136	[X.509]	International Telecommunication Union (ITU)–T, X.509: Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks, August 2005, http://www.itu.int/rec/T-REC-X.509-200508- l/en
137 138	[X9.24-1]	ANSI, X9.24: Retail Financial Services Symmetric Key Management - Part 1: Using Symmetric Techniques, 2004.
139 140	[X9.31]	ANSI, X9.31: Digital Signatures Using Reversible Public Key Cryptography for the Financial Services Industry (rDSA), September 1998.
141 142	[X9.42]	ANSI, X9-42: Public Key Cryptography for the Financial Services Industry: Agreement of Symmetric Keys Using Discrete Logarithm Cryptography, 2003.
143 144	[X9-57]	ANSI, X9-57: Public Key Cryptography for the Financial Services Industry: Certificate Management 1997
145	[X9.62]	ANSI, X9-62: Public Key Cryptography for the Financial Services Industry. The
146	[]	Elliptic Curve Digital Signature Algorithm (ECDSA), 2005.
147	[X9-63]	ANSI, X9-63: Public Key Cryptography for the Financial Services Industry, Key
148		Agreement and Key Transport Using Elliptic Curve Cryptography, 2001.
149 150	[X9-102]	ANSI, X9-102: Symmetric Key Cryptography for the Financial Services Industry - Wrapping of Keys and Associated Data, 2008.

Field Code Changed

kmip-ug-1.0-cd-<u>09</u> Copyright © OASIS® <u>2010.</u>2009. All Rights Reserved.

151[X9 TR-31]ANSI, X9 TR-31: Interoperable Secure Key Exchange Key Block Specification for
Symmetric Algorithms, 2005.

153 **1.3 Non-normative References**

154	[KMIP-UC]	OASIS Committee Draft 0905, Key Management Interoperability Protocol Use		
155		Cases Version 1.0, March 2010November 2009. http://docs.oasis-	(Formatted: Hyperlink
156		open.org/kmip/usecases/v1.0/ <u>cd09</u> cd05/kmip-usecases-1.0-cd- <u>09</u> 05.doc	(Field Code Changed
157		4.	{	Formatted: Ref, Indent: First line: 0 pt

kmip-ug-1.0-cd-<u>09</u> Copyright © OASIS® <u>2010,</u>2009. All Rights Reserved.

158 2 Assumptions

The section describes assumptions that underlie the KMIP protocol and the implementation of clients and servers that utilize the protocol.

161 2.1 Island of Trust

162 Clients may be provided key material by the server, but they only use that keying material for the 163 purposes explicitly listed in the delivery payload. Clients that ignore these instructions and use the keys in 164 ways not explicitly allowed by the server are non-compliant. There is no requirement for the key

165 management system, however, to enforce this behavior.

166 2.2 Message Security

167 KMIP relies on the chosen authentication suite as specified in [KMIP-Prof] to authenticate the client and 168 on the underlying transport protocol to provide confidentiality, integrity, message authentication and

protection against replay attack. KMIP offers a wrapping mechanism for the Key Value that does not rely on the transport mechanism used for the messages; the wrapping mechanism is intended for importing or exporting managed cryptographic objects.

172 2.3 State-less Server

173 The protocol operates on the assumption that the server is state-less, which means that there is no

174 concept of "sessions" inherent in the protocol. State-less server operation is much more reliable and

175 easier to implement than stateful operation, and is consistent with possible implementation scenarios,

such as web-services-based servers. This does not mean that the server itself maintains no state, only

177 that the protocol does not require this.

178 2.4 Extensible Protocol

The protocol provides for "private" or vendor-specific extensions, which allow for differentiation among
 vendor implementations. However, any objects, attributes and operations included in an implementation
 are always implemented as specified in [KMIP-Spec], regardless of whether they are optional or
 mandatory.

183 2.5 Server Policy

A server is required to be conformant to KMIP and support the conformance clauses as specified in
 [KMIP-Spec]. However, a server may refuse a server-supported operation or client-settable attribute if
 disallowed by the server policy.

187 2.52.6 Support for Cryptographic Objects

188 The protocol supports all reasonable key management system-related cryptographic objects. This list 189 currently includes:

- 190 Symmetric Keys
- 191 Split (multi-part) Keys
- 192 Asymmetric Key Pairs and their components
- 193 Digital Certificates
- 194 Derived Keys
 - Secret Data

Opaque (non-interpretable) cryptographic objects

kmip-ug-1.0-cd-<mark>09</mark>

195

196

Copyright © OASIS® 2010,2009. All Rights Reserved.

18 March 201005 Page 11 of 46

197 2.62.7 Client-Server Message-based Model

The protocol operates primarily in a client-server, message-based model. This means that most protocol 198 199 exchanges are initiated by a client sending a request message to a server, which then sends a response 200 to the client. The protocol also provides optional mechanisms to allow for unsolicited notification of events 201 to clients using the Notify operation, and unsolicited delivery of cryptographic objects to clients using the 202 Put operation; that is, the protocol allows a "push" model, whereby the server initiates the protocol 203 exchange with either a Notify or Put operation. These Notify or Put features are optionally supported by servers and clients. Clients may register in order to receive such events/notifications. Registration is 204 205 implementation-specific and not described in the specification.

206 2.72.8 Synchronous and Asynchronous Messages

The protocol allows two modes of operation. Synchronous (mandatory) operations are those in which a
client sends a request and waits for a response from the server. Polled Asynchronous operations
(optional) are those in which the client sends a request, the server responds with a "pending" status, and
the client polls the server for the completed response and completion status. Server implementations may
choose not to support the Polled Asynchronous feature of the protocol.

212 2.82.9 Support for "Intelligent Clients" and "Key Using Devices"

The protocol supports intelligent clients, such as end-user workstations, which are capable of requesting
 all of the functions of KMIP. It also allows subsets of the protocol and possible alternate message
 representations in order to support less-capable devices, which only need a subset of the features of
 KMIP.

217 **2.92.10** Batched Requests and Responses

218 The protocol contains a mechanism for sending batched requests and receiving the corresponding 219 batched responses, to allow for higher throughput on operations that deal with a large number of entities, 220 e. g., requesting dozens or hundreds of keys from a server at one time, and performing operations in a 221 group. An option is provided to indicate whether to continue processing requests after an earlier request 222 in the batch fails or to stop processing the remaining requests in the batch. Note that there is no option to 223 treat an entire batch as atomic, that is, if a request in the batch fails, then preceding requests in the batch 224 are not undone or rolled back (see Section 3.15). A special ID Placeholder (see Section 3.19) is provided 225 in KMIP to allow related requests in a batch to be pipelined.

226 2.102.11 Reliable Message Delivery

The reliable message delivery function is relegated to the transport protocol, and is not part of the key management protocol itself.

229 2.112 Large Responses

For requests that could result in large responses, a mechanism in the protocol allows a client to specify in a request the maximum allowed size of a response. The server indicates in a response to such a request that the response would have been too large and, therefore, is not returned.

233 2.122.13 Key Life-cycle and Key State

[KMIP-Spec] describes the key life-cycle model, based on the NIST SP 800-57 key state definitions
 [SP800-57-1], supported by the KMIP protocol. Particular implications of the key life-cycle model in terms
 of defining time-related attributes of objects are discussed in Section 3.5_below.

237

kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved.

238 **3 Usage Guidelines**

This section provides guidance on using the functionality described in the Key ManagementInteroperability Protocol Specification.

241 3.1 Authentication

As discussed in [KMIP-Spec], a conforming KMIP implementation establishes and maintains channel
 confidentiality and integrity, and provides assurance of proves server authenticity for KMIP messaging.
 Client authentication is performed according to the chosen KMIP authentication suite as specified in
 [KMIP-Prof]. Other mechanisms for client and server authentication are possible and optional for KMIP
 implementations.

KMIP implementations that support the KMIP-defined Credential Types or use other vendor-specific 247 mechanisms for authentication may use the optional Authentication field specified inside the Request 248 249 HeaderCredential attribute to include additional identification information. Depending on the server's 250 configuration, the server may interpret the identity of the requestor from the Credential object, contained 251 in the Authentication structure if it is not provided during the channel-level authentication. For example, in 252 addition to performing mutual authentication during a SSL/TLS handshake, the client passes the 253 Credential object (e.g., a username and password) in the request. If the requestor's username is not specified inside the client certificate and is instead specified in the Credential object, the server interprets 254 255 the identity of the requestor from the Credential object. This supports use cases where channel-level authentication authenticates a machine or service that is used by multiple users of the KMIP server. If the 256 257 client provides the username of the requestor in both the client certificate and the Credential object, the 258 server verifies that the usernames are the same. If they differ, the authentication fails and the server 259 returns an error. If no Credential object is included in the request, the username of the requestor is 260 expected to be provided inside the certificate. If no username is provided in the client certificate and no Credential object is included in the request message, the server is expected to refuse authentication and 261 262 return an error.

If authentication is unsuccessful, and it is possible to return an "authentication not successful" error, this error should be returned in preference to any other result status. This prevents status code probing by a client that is not able to authenticate.

Server decisions regarding which operations to reject if there is insufficiently strong authentication of the client are not specified in the protocol. However, see Section 3.2 for operations for which authentication and authorization are particularly important.

269 3.1.1 Credential

270 [KMIP-Spec] defines the Username and Password structure for the Credential Type Username and 271 Password. The structure consists of two fields: Username and Password. Password is a recommended, 272 but optional, field, which may be excluded only if the client is authenticated using one of the 273 authentication suites defined in [KMIP-Prof]. For example, if the client performs client certificate 274 authentication during the TLS handshake, and the Authentication field is provided in the Message 275 Request, the Password field is an optional field in the Username and Password structure of the Credential 276 object. 277 278 The Credential object is used to provide additional identification information. As described above, for 279 certain use cases, channel-level authentication may only authenticate a machine or service that is used 280 by multiple clients of the KMIP server. The Credential object may be used in this scenario to identify

<u>individual clients of the Kinin server. The credential object may be used in this scenario to identify</u>
 <u>individual clients by specifying the username in the Username and Password structure. Depending on the</u>
 <u>client's environment, the username may be the device's serial number, the volume name or some other</u>

283 unique identifier.

kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved. 18 March 201005 Page 13 of 46 Formatted: Font:

Multiple clients should not be authenticated using the same channel-level authentication credential (e.g.,
 the same client certificate). The Credential object may be used to authenticate individual clients by
 requiring the Username and Password to be provided in the Credential object.

Authorization for Revoke, Recover, Destroy and Archive Operations

Neither authentication nor authorization is handled by the KMIP protocol directly. In particular, the
 Credential attribute is not guaranteed to be an authenticated identity of the requesting client. However,
 the authentication suite, as specified in [KMIP-Prof], describes how the client identity is established for

KMIP-compliant implementations. This authentication is performed for all KMIP operations, with the single exception of the Query operation.

Certain operations that may be requested by a client via KMIP, particularly Revoke, Recover, Destroy and
Archive, may have a significant impact on the availability of a key, on server performance and on key
security. When a server receives a request for one of these operations, it should ensure that the client
has authenticated its identity (see the Authentication Suites section in [KMIP-Prof]). The server should
also ensure that the client requesting the operation is an object creator, security officer or other identity
authorized to issue the request. It may also require additional authentication to ensure that the object

300 owner or a security officer has issued that request. Even with such authentication and authorization,

requests for these operations should be considered only a "hint" to the key management system, which

302 may or may not choose to act upon this request.

303 3.3 Using Notify and Put Operations

The Notify and Put operations are the only operations in the KMIP protocol that are initiated by the server, rather than the client. As client-initiated requests are able to perform these functions (e.g., by polling to request notification), these operations are optional for conforming KMIP implementations. However, they provide a mechanism for optimized communication between KMIP servers and clients and have,

308 therefore, been included in [KMIP-Spec].

309 In using Notify and Put, the following constraints and guidelines should be observed:

- The client registers with the server, so that the server knows how to locate the client to which a
 Notify or Put is being sent and which events for the Notify are supported. However, such
 registration is outside the scope of the KMIP protocol. Registration also includes a specification of
 whether a given client supports Put and Notify, and what attributes may be included in a Put for a
 particular client.
- Communication between the client and the server is properly authenticated to forestall man-inthe-middle attacks in which the client receives Notify or Put operations from an unauthenticated server. Authentication for a particular client/server implementation is at a minimum accomplished using one of the mandatory authentication mechanisms (see [KMIP-Prof]). Further strengthening of the client/server communications integrity by means of signed message content and/or wrapped keys is recommended. Attribute values other than "Last Change Date" should not be included in a Notify to minimize risk of exposure of attribute information.
- In order to minimize possible divergence of key or state information between client and server as a result of server-initiated communication, any client receiving Notify or Put messages returns acknowledgements of these messages to the server. This acknowledgement may be at communication layers below the KMIP layer, such as by using transport-level acknowledgement provided in TCP/IP.
- For client devices that are incapable of responding to messages from the server, communication with the server happens via a proxy entity that communicates with the server, using KMIP, on behalf of the client. It is possible to secure communication between a proxy entity and the client using other, potentially proprietary mechanisms.

331 3.4 Usage Allocation

Usage should be allocated and handled carefully at the client, since power outages or other types of 332 333 client failures (crashes) may render allocated usage lost. For example, in the case of a key being used for 334 the encryption of tapes, such a loss of the usage allocation information following a client failure during 335 encryption may result in the necessity for the entire tape backup session to be re-encrypted using a 336 different key, if the server is not able to allocate more usage. It is possible to address this through such 337 approaches as caching usage allocation information on stable storage at the client, and/or having conservative allocation policies at the server (e.g., by keeping the maximum possible usage allocation per 338 339 client request moderate). In general, usage allocations should be as small as possible; it is preferable to 340 use multiple smaller allocation requests rather than a single larger request to minimize the likelihood of 341 unused allocation.

342 3.5 Key State and Times

343 [KMIP-Spec] provides a number of time-related attributes, including the following:
 344 Initial Date: The date and time when the managed cryptographic object was first of

- Initial Date: The date and time when the managed cryptographic object was first created by or registered at the server
- Activation Date: The date and time when the managed cryptographic object may begin to be used for applying cryptographic protection to data
- Process Start Date: The date and time when a managed symmetric key object may begin to be used for processing cryptographically protected data
- Protect Stop Date: The date and time when a managed symmetric key object may no longer be used for applying cryptographic protection to data
- Deactivation Date: The date and time when the managed cryptographic object may no longer be
 used for any purpose, except for decryption, signature verification, or unwrapping, but only under
 extraordinary circumstances and when special permission is granted
- Destroy Date: The date and time when the managed cryptographic object was destroyed
- Compromise Occurrence Date: The date and time when the managed cryptographic object was first believed to be compromised
- Compromise Date: The date and time when the managed cryptographic object is entered into the compromised state
 - Archive Date: The date and time when the managed object was placed in Off-Line storage

These attributes apply to all cryptographic objects (symmetric keys, asymmetric keys, etc) with exceptions as noted in [KMIP-Spec]. However, certain of these attributes (such as the Initial Date) are not specified by the client and are implicitly set by the server.

- In using these attributes, the following guidelines should be observed:
- 365 As discussed for each of these attributes in Section 3 of [KMIP-Spec], a number of these times 366 are set once and it is not possible for the client or server to modify them. However, several of the time attributes (particularly the Activation Date, Protect Start Date, Process Stop Date and 367 368 Deactivation Date) may be set by the server and/or requested by the client. Coordination of timerelated attributes between client and server, therefore, is primarily the responsibility of the server, 369 370 as it manages the cryptographic object and its state. However, special conditions related to timerelated attributes, governing when the server accepts client modifications to time-related 371 372 attributes, may be negotiated by policy exchange between the client and server, outside the Key 373 Management Interoperability Protocol. 374 In general, state transitions occur as a result of operational requests, such as Create, Create Key 375 376 Pair, Register, Activate, Revoke, and Destroy. However, clients may need to specify times in the future for such things as Activation Date, Deactivation Date, Process Start Date, and Protect Stop 377 378 Date.
- 379

360

380	KMIP allows clients to specify times in the past for such attributes as Activation Date and
381	Deactivation Date. This is intended primarily for clients that were disconnected from the server at
382	the time that the client performed that operation on a given key.

- It is valid to have a projected Deactivation Date when there is no Activation Date. This means, however, that the key is not yet active, even though its projected Deactivation Date has been specified. A valid Deactivation Date is greater than or equal to the Activation Date.
- The Protect Stop Date may be equal to, but may not be later than the Deactivation Date.
 Similarly, the Process Start Date may be equal to, but may not precede, the Activation Date.
 KMIP implementations should consider specifying both these attributes, particularly for symmetric keys, as a key may be needed for processing protected data (e.g., decryption) long after it is no longer appropriate to use it for applying cryptographic protection to data (e.g., encryption).
- KMIP does not allow an Active object to be destroyed with the Destroy operation. The server is required to return an error, if the client invokes the Destroy operation on an Active object. To destroy an Active object, clients are required to first call the Revoke operation or explicitly set the Deactivation Date of the object. Once the object is in Deactivated state, clients may destroy the object by calling the Destroy operation. These operations may be performed in a batch. If other time-related attributes (e.g., Protect Stop Date) are set to a future date, the server should set these to the Deactivation Date.
- If a Destroy operation is performed, resulting in the Destroy Date being set, and the object has
 not already been deactivated, the deactivation of the object is also performed prior to the Destroy
 operation, so that Destroy Date is greater than or equal to the Deactivation Date. If other time related attributes (e.g., Protect Stop Date) are set to a future date, the server should set these to
 the deactivation date.
 - After a cryptographic object is destroyed, a key management server may retain certain information about the object, such as the Unique Identifier.

KMIP allows the specification of attributes on a per-client basis, such that a server could maintain or
present different sets of attributes for different clients. This flexibility may be necessary in some cases,
such as when a server maintains the availability of a given key for some clients, even after that same key
is moved to an inactive state (e.g., <u>Deactivated</u>, <u>deactivated</u> state) for other clients. However, such an
approach might result in significant inconsistencies regarding the object state from the point of view of all
participating clients and should, therefore, be avoided. A server should maintain a consistent state for
each object, across all clients that have or are able to request that object.

412 3.6 Template

403

404

413	The usage of templates is an alternative approach for setting attributes in an operation request. Instead of		
414 415	individually specifying each attribute, a template may be used to set any of the following attributes for a managed object:		
416	Cryptographic Algorithm		
417	Cryptographic Length		
418	Cryptographic Domain Parameters		
419	Cryptographic Parameters		
420	Operation Policy Name		
421	Cryptographic Usage Mask		
422	Usage Limits		
423	Activation Date		
424	Process Start Date		
425	Protect Stop Date		
426	Deactivation Date		
427	Object Group		
	kmip-ug-1.0-cd-09 18 March 2010 05		

5 November 2009

Copyright © OASIS® 2010.2009. All Rights Reserved.

Page 16 of 46

<u>Custom Attribute</u>		
In addition to these attributes, the template has attributes that are appl	licable to the template itself. These	
include the attributes (Unique Identifier, Initial Date, Last Change Date	e, and Archive Date) set implicitly	
after successfully completing a certain operation and attributes set by in the Register request. When registering a template, the Name attribu-	the client (Object Type and Name) ite for the template should be set. It	
is used to specify and identify the template in the Template-Attribute s	tructure when attributes for a	
managed object are set.		
The Template-Attribute structure allows for multiple template names a	nd individual attributes to be	
specified in an operation request. The structure is used in the Create,	<u>Create Key Pair, Register, Re-key,</u>	
Derive Key, Certify, and Re-certify operations. All of these operations	with the exception of the Create	
Template-Attribute Private Key Template Attribute tag. The Create Key Pa	mplate-Attribute tags	
remplate Attribute, Findle Rey Femplate Attribute, and Fabile Rey Fe		
Templates may be the subject of the Register, Locate, Get. Get Attribu	utes, Get Attribute List, Add	
Attribute, Modify Attribute, Delete Attribute, Delete Attribute, and Destr	oy operations. Clients are not able	
to create a template with the Create operation; instead templates are operation. When the template is the subject of the operation, the University of the operation.	created using the Register	
template. The template name is only used to identify the template inside	de a Template-Attribute structure.	
3.6.1.1 Example of Registering a Template n this example, a client registers a template by encapsulating attribute	es for creating a 256-bit AES key	
with the Cryptographic Usage Mask set to Encrypt and Decrypt.		
The following is specified inside the Register Request Payload:		
Object Type: Template		
Template-Attribute:		
– Name: Template1		
 Cryptographic Algorithm: AES 		
 Cryptographic Length: 256 		
 Cryptographic Usage Mask: Encrypt and Decrypt 		
 Cryptographic Usage Mask: Encrypt and Decrypt Operation Policy Name: OperationPolicy1 		
 Cryptographic Usage Mask: Encrypt and Decrypt Operation Policy Name: OperationPolicy1 The Operation Policy OperationPolicy1 applies to the AES key being operation Policy 1 applies to the AES key being operation 	created using the template. It is not	
 Cryptographic Usage Mask: Encrypt and Decrypt Operation Policy Name: OperationPolicy1 The Operation Policy OperationPolicy1 applies to the AES key being or used to control operations on the template itself. KMIP does not allow for controlling operations on the template itself. The default policy for the controlling operations on the template itself. 	created using the template. It is not operation policies to be specified	
 Cryptographic Usage Mask: Encrypt and Decrypt Operation Policy Name: OperationPolicy1 The Operation Policy OperationPolicy1 applies to the AES key being of used to control operations on the template itself. KMIP does not allow for controlling operations on the template itself. The default policy for the purpose and is specified in the KMIP Specification. 	created using the template. It is not operation policies to be specified emplate objects is used for this	
 <u>Cryptographic Usage Mask: Encrypt and Decrypt</u> <u>Operation Policy Name: OperationPolicy1</u> <u>The Operation Policy OperationPolicy1 applies to the AES key being coused to control operations on the template itself. KMIP does not allow for controlling operations on the template itself. The default policy for the purpose and is specified in the KMIP Specification.</u> 3.6.1.2 Example of Creating a Symmetric Key usi 	preated using the template. It is not operation policies to be specified emplate objects is used for this ng a Template	
Cryptographic Usage Mask: Encrypt and Decrypt Operation Policy Name: OperationPolicy1 The Operation Policy OperationPolicy1 applies to the AES key being of used to control operations on the template itself. KMIP does not allow for controlling operations on the template itself. The default policy for to purpose and is specified in the KMIP Specification. S.6.1.2 Example of Creating a Symmetric Key usi In this example, the client uses the template created in example 3.6.1	created using the template. It is not operation policies to be specified emplate objects is used for this ng a Template to create a 256-bit AES key.	
Cryptographic Usage Mask: Encrypt and Decrypt Operation Policy Name: OperationPolicy1 The Operation Policy OperationPolicy1 applies to the AES key being of used to control operations on the template itself. KMIP does not allow for controlling operations on the template itself. The default policy for to purpose and is specified in the KMIP Specification. S.6.1.2 Example of Creating a Symmetric Key usi In this example, the client uses the template created in example 3.6.1	created using the template. It is not operation policies to be specified emplate objects is used for this ng a Template to create a 256-bit AES key.	
Cryptographic Usage Mask: Encrypt and Decrypt Operation Policy Name: OperationPolicy1 The Operation Policy OperationPolicy1 applies to the AES key being of used to control operations on the template itself. KMIP does not allow for controlling operations on the template itself. The default policy for to purpose and is specified in the KMIP Specification. Scheduler Statement Statement Statement Statement Statement Statement In this example, the client uses the template created in example 3.6.1 The following is specified in the Create Request Pavload:	ereated using the template. It is not operation policies to be specified emplate objects is used for this ng a Template to create a 256-bit AES key.	
Cryptographic Usage Mask: Encrypt and Decrypt Operation Policy Name: OperationPolicy1 The Operation Policy OperationPolicy1 applies to the AES key being of used to control operations on the template itself. KMIP does not allow for controlling operations on the template itself. The default policy for the purpose and is specified in the KMIP Specification. 3.6.1.2 Example of Creating a Symmetric Key usi In this example, the client uses the template created in example 3.6.1 The following is specified in the Create Request Payload: kmip-ug-1.0-cd-09	ereated using the template. It is not operation policies to be specified emplate objects is used for this ng a Template to create a 256-bit AES key.	

473 474 Object Type: Symmetric Key 475 Template-Attribute: 476 Name: Template1 477 Attribute: 478 Name: AESkey 479 Custom Attribute: x-ID74592 480 481 The Template-Attribute specifies both a template name and additional attributes. The Name attribute is 482 not an attribute that may be set by a template. The Name attribute set for the template applies to the template itself (e.g., Template1 is the Name attribute of the Template object). The Name attribute for the 483 484 symmetric key is therefore specified separately under Attribute. It is possible to specify the Custom 485 Attribute inside the template; however, this particular example sets this attribute separately.

486 It is possible for a server to maintain different policy templates for different clients. As in the state
 487 transitions described above, however, this practice is discouraged.

488 3.7 Archive Operations

When the Archive operation is performed, it is recommended that an object identifier and a minimal set of
attributes be retained within the server for operational efficiency. In such a case, the retained attributes
may include Unique Identifier and State.

492 **3.8 Message Extensions**

Any number of vendor-specific extensions may be included in the Message Extension optional structure.
 This allows KMIP implementations to create multiple extensions to the protocol.

495 3.9 Unique Identifiers

496 For clients that require unique identifiers in a special form, out-of-band registration/configuration may be
 497 used to communicate this requirement to the server.

498 3.10 Result Message Text

499 KMIP specifies the Result Status, the Result Reason and the Result Message as normative message

- 500 contents. For the Result Status and Result Reason, the enumerations provided in [KMIP-Spec] are the 501 normative values. The values for the Result Message text, on the other hand, are implementation-
- 501 specific. In consideration of internationalization, it is recommended that any vendor implementation of
- 502 Specific. In Consideration of Internationalization, it is recommended that any vehicle implementation of 503 KMIP provide appropriate language support for the Return Message. How a client specifies the language 504 for Result Messages is outside the scope of the KMIP.

505 3.11 Query

506 Query does not explicitly support client requests to determine what operations require authentication. To 507 determine whether an operation requires authentication, a client should request that operation.

508 3.12 Canceling Asynchronous Operations

- 509 If an asynchronous operation is cancelled by the client, no information is returned by the server in the
- 510 result code regarding any operations that may have been partially completed. Identification and 511 remediation of partially completed operations is the responsibility of the server.

- 512 It is the responsibility of the server to determine when to discard the status of asynchronous operations.
- 513 The determination of how long a server should retain the status of an asynchronous operation is
- 514 implementation-dependent and not defined by KMIP.
- 515 Once a client has received the status on an asynchronous operation other than "pending", any
- subsequent request for status of that operation may return either the same status as in a previous polling 516 request or an "unavailable" response. 517

3.13 Multi-instance Hash 518

519 The Digest attribute contains the output of hashing a managed object, such as a key or a certificate. The server always generates the SHA-256 hash value when the object is created or generated. KMIP allows 520 521 multiple instances of the digest attribute to be associated with the same managed object. For example, it is common practice for publicly trusted CAs to publish two digests (often referred to as the fingerprint or 522 523 the thumbprint) of their certificate: one calculated using the SHA-1 algorithm and another using the MD-5

524 algorithm. In this case, each digest would be calculated by the server using a different hash algorithm.

3.14 Returning Related Objects 525

526 The key block is intended to return a single object, with associated attributes and other data. For those 527 cases in which multiple related objects are needed by a client, such as the private key and the related certificate specified by RACF and JKS, the client should issue multiple Get requests to obtain these 528 529 related objects.

3.15 Reducing Multiple Requests through the Use of Batch 530

531 KMIP supports batch operations in order to reduce the number of calls between the client and server for related operations. For example, Locate and Get are likely to be commonly accomplished within a single 532 533 batch request.

534 KMIP does not ensure that batch operations are atomic on the server side. If servers implement such atomicity, the client is able to use the optional "undo" mode to request roll-back for batch operations 535 implemented as atomic transactions. However, support for "undo" mode is optional in the protocol, and 536 there is no guarantee that a server that supports "undo" mode has effectively implemented atomic 537 batches. The use of "undo", therefore, should be restricted to those cases in which it is possible to assure 538 the client, through mechanisms outside of KMIP, of the server effectively supporting atomicity for batch 539 540 operations.

3.16 Maximum Message Size 541

542 When a server is processing requests in a batch, it should compare the cumulative response size of the

- 543 message to be returned after each request with the specified Maximum Response Size. If the message is too large, it should prepare a maximum message size error response message at that point, rather than 544
- 545 continuing with operations in the batch. This increases the client's ability to understand what operations
- 546 have and have not been completed.
- 547 When processing individual requests within the batch, the server that has encountered a Maximum 548 Response Size error should not return attribute values or other information as part of the error response.

3.17 Using Offset in Re-key and Re-certify Operations 549

- 550 Both the Re-key and the Re-certify operations allow the specification of an offset interval.
- The Re-key operation allows the client to specify an offset interval for activation of the key. This offset 551
- specifies the duration of time between the time the request is made and the time when the activation of 552
- 553 the key occurs. If an offset is specified, all other times for the new key are determined from the new
- Activation Date, based on the intervals used by the previous key, i.e., from the Activation Date to the 554
- Process Start Date, Protect Stop Date, etc. 555

kmip-ua-1.0-cd-09 Copyright © OASIS® 2010, 2009. All Rights Reserved. 556 The Re-certify operation allows the client to specify an offset interval that indicates the difference between

- 557 the Initial Date of the new certificate and the Activation Date of the new certificate. As with the Re-key
- 558 operation, all other times for the certificate are determined using the intervals used for the previous 559 certificate.

560 **3.18 Locate Queries**

- 561 It is possible to formulate Locate queries to address any of the following conditions:
- Exact match of a transition to a given state. Locate the key(s) with a transition to a certain state at a specified time (t).
- Range match of a transition to a given state. Locate the key(s) with a transition to a certain state at any time at or between two specified times (t and t').
- Exact match of a state at a specified time. Locate the key(s) that are in a certain state at a specified time (t).
- Match of a state during an entire time range. Locate the key(s) that are in a certain state during
 an entire time specified with times (t and t'). Note that the Activation Date could occur at or before
 t and that the Deactivation Date could occur at or after t'+1.
- Match of a state at some point during a time range. Locate the key(s) that are in a certain state at some time at or between two specified times (t and t'). In this case, the transition to that state could be before the start of the specified time range.
- This is accomplished by allowing any date/time attribute to be present either once (for an exact match) or at most twice (for a range match).
- 576 For instance, if the state we are interested in is Active, the Locate queries would be the following 577 (corresponding to the bulleted list above):
- Exact match of a transition to a given state: Locate (ActivationDate(t)). Locate keys with an
 Activation Date of t.
- Range match of a transition to a given state: Locate (ActivationDate(t), ActivationDate(t')). Locate keys with an Activation Date at or between t and t'.
- Exact match of a state at a specified time: Locate (ActivationDate(0), ActivationDate(t), DeactivationDate(t+1), DeactivationDate(MAX_INT), CompromiseDate(t+1), CompromiseDate(MAX_INT)). Locate keys in the Active state at time t, by looking for keys with a transition to Active before or until t, and a transition to Deactivated or Compromised after t (because we don't want the keys that have a transition to Deactivated or Compromised before t). The server assumes that keys without a DeactivationDate or CompromiseDate is equivalent to MAX_INT (i.e., infinite).
- Match of a state during an entire time range: Locate (ActivationDate(0), ActivationDate(t),
 DeactivationDate(t'+1), DeactivationDate(MAX_INT), CompromiseDate(t'+1),
 CompromiseDate(MAX_INT)). Locate keys in the Active state during the entire time from t to t'.
- Match of a state at some point during a time range: Locate (ActivationDate(0), ActivationDate(t'-1), DeactivationDate(t+1), DeactivationDate(MAX_INT), CompromiseDate(t+1), CompromiseDate(MAX_INT)). Locate keys in the Active state at some time from t to t', by looking for keys with a transition to Active between 0 and t'-1 and exit out of Active on or after t+1.
- 596 The gueries would be similar for Initial Date, Deactivation Date, Compromise Date and Destroy Date.
- 597 In the case of the Destroyed-Compromise state, there are two dates recorded: the Destroy Date and the 598 Compromise Date. For this state, the Locate operation would be expressed as follows:
- Exact match of a transition to a given state: Locate (CompromiseDate(t), State(Destroyed-Compromised)) and Locate (DestroyDate(t), State(Destroyed-Compromised)). KMIP does not support the OR in the Locate request, so two requests should be issued. Locate keys that were Destroyed and transitioned to the Destroyed-Compromised state at time t, and locate keys that were Compromised and transitioned to the Destroyed-Compromised state at time t.

- Range match of a transition to a given state: Locate (CompromiseDate(t), CompromiseDate(t'), State(Destroyed-Compromised)) and Locate (DestroyDate(t), DestroyDate(t'), State(Destroyed-Compromised)). Locate keys that are Destroyed-Compromised and were Compromised or Destroyed at or between t and t'.
- Exact match of a state at a specified time: Locate (CompromiseDate(0), CompromiseDate(t), DestroyDate(0), DestroyDate(t)); nothing else is needed, since there is no exit transition. Locate keys with a Compromise Date at or before t, and with a Destroy Date at or before t. These keys are, therefore, in the Destroyed-Compromised state at time t.
- Match of a state during an entire time range: Locate (CompromiseDate(0), CompromiseDate(t),
 DestroyDate(0), DestroyDate(t)). Same as above. As there is no exit transition from the
 Destroyed-Compromised state, the end of the range (t') is irrelevant.
- Match of a state at some point during a time range: Locate (CompromiseDate(0),
 CompromiseDate(t'-1), DestroyDate(0), DestroyDate(t'-1)). Locate keys with a Compromise Date
 at or before t'-1, and with a Destroy Date at or before t'-1. As there is no exit transition from the
 Destroyed-Compromised state, the start of the range (t) is irrelevant.

619 3.19 ID Placeholder

- 620 A number of operations are affected by a mechanism referred to as the ID Placeholder. This is a
- 621 temporary variable consisting of a single Unique Identifier that is stored inside the server for the duration
- 622 of executing a batch of operations. The ID Placeholder is obtained from the Unique Identifier returned by
- 623 certain operations; the applicable operations are identified in <u>Table 1-Table 1</u>, along with a list of
- 624 operations that accept the ID Placeholder as input.

Operation	ID Placeholder at the beginning of the operation	ID Placeholder upon completion of the operation (in case of operation failure, a batch using the ID Placeholder stops)
Create	-	ID of new Object
Create Key Pair	-	ID of new Private Key (ID of new Public Key may be obtained via a Locate)
Register	-	ID of newly registered Object
Derive Key	- (multiple Unique Identifiers may be specified in the request)	ID of new Symmetric Key
Locate	-	ID of located Object
Get	ID of Object	no change
Request Object	I D of Object	no change
Validate	-	-
Get Attributes List/Modify/Add/Delete	ID of Object	no change
Activate	ID of Object	no change
Revoke	ID of Object	no change
Destroy	ID of Object	no change
Archive/Recover	ID of Object	no change

kmip-ug-1.0-cd-09

Copyright © OASIS® 2010.2009. All Rights Reserved.

Certify	ID of Public Key	ID of new Certificate
Re-certify	ID of Certificate	ID of new Certificate
Re-key	ID of Symmetric Key to be rekeyed	ID of new Symmetric Key
Obtain Lease	ID of Object	no change
Get Usage Allocation	ID of Key	no change
<u>Check</u>	ID of Object	no change

625

638

639

646

647

648

649

650

651

652 653

655

Table 1: ID Placeholder Prior to and Resulting from a KMIP Operation

626 3.20 Key Block

The protocol uses the Key Block structure to transport a key to the client or server. This Key Block consists of the Key Value Type, the Key Value, and the Key Wrapping Data. The Key Value Type identifies the format of the Key Material, e.g., Raw format or Transparent Key structure. The Key Value consists of the Key Material and optional attributes. The Key Wrapping Data provides information about the wrapping key and the wrapping mechanism, and is returned only if the client requests the Key Value to be wrapped by specifying the Key Wrapping Specification inside the Get Request Payload. The Key Wrapping Data may also be included inside the Key Block if the client registers a wrapped key.

The protocol allows any attribute to be included inside the Key Value and allows these attributes to be cryptographically bound to the Key Material (i.e., by signing, MACing, encrypting, or both encrypting and signing/MACing the Key Value). Some of the attributes that may be included include the following:

- Onique Identifier uniquely identifies the key
 - Cryptographic Algorithm (e.g., AES, 3DES, RSA) this attribute is either specified inside the Key Block structure or the Key Value structure-
- 640
 Cryptographic Length (e.g., 128, 256, 2048) this attribute is either specified inside the Key Block structure or the Key Value structure
- 642 Cryptographic Usage Mask- identifies the cryptographic usage of the key (e.g., Encrypt, Wrap
 643 Key, Export)
- 644
 Cryptographic Parameters provides additional parameters for determining how the key may be used
 - Block Cipher Mode (e.g., CBC, NISTKeyWrap, GCM) this parameter identifies the mode of operation, including block cipher-based MACs or wrapping mechanisms
 - Padding Method (e.g., OAEP, X9.31, PSS) identifies the padding method and if applicable the signature or encryption scheme.
 - Hashing Algorithm (e.g., SHA-256) identifies the hash algorithm to be used with the signature/encryption mechanism or Mask Generation Function; note that the different HMACs are defined individually as algorithms and do not require the Hashing Algorithm parameter to be set
- 654 <u>Key</u> Role Type Identifies the financial key role (e.g., DEK, KEK)
 - State (e.g., Active)
- Dates (e.g., Activation Date, Process Start Date, Protect Stop Date)
- Custom Attribute allows vendors and clients to define vendor-specific attributes; may also be used to prevent replay attacks by setting a nonce

659 3.21 Using Wrapped Keys with KMIP

660 KMIP provides the option to register and get keys in wrapped format. Clients request the server to return 661 a wrapped key by including the Key Wrapping Specification in the Get Request Payload. Similarly, clients register a wrapped key by including the Key Wrapping Data in the Register Request Payload. The 662 663 Wrapping Method identifies the type of mechanism used to wrap the key, but does not identify the 664 algorithm or block cipher mode. It is possible to determine these from the attributes set for the specified 665 Encryption Key or MAC/Signing Key. If a key has multiple Cryptographic Parameters set, clients may include the applicable parameters in Key Wrapping Specification. If omitted, the server chooses the 666 667 Cryptographic Parameter attribute with the lowest index.

The Key Value includes both the Key Material and, optionally, attributes of the key; these may be provided by the client in the Register Request Payload; the server only includes attributes when requested in the Key Wrapping Specification of the Get Request Payload. The Key Value may be

671 encrypted, signed/MACed, or both encrypted and signed/MACed (and vice versa). In addition, clients

672 have the option to request or import a wrapped Key Block according to standards, such as ANSI TR-31, 673 or vendor-specific key wrapping methods.

673 or vendor-specific key wrapping methods.

It is important to note that if the Key Wrapping Specification is included in the Get Request Payload, the Key Value may not necessarily be encrypted. If the Wrapping Method is MAC/sign, the returned Key Value is in plaintext, and the Key Wrapping Date includes the MAC or Signature of the Key Value

Value is in plaintext, and the Key Wrapping Data includes the MAC or Signature of the Key Value.

Prior to wrapping or unwrapping a key, the server should verify that the wrapping key is allowed to be
used for the specified purpose. For example, if the Unique ID of a symmetric key is specified used for key
encryption in the Key Wrapping Specification inside the response to a Get request, the symmetric key
should have the "Wrap Key" bit set in its Cryptographic Usage Mask. Similarly, if the client registers a
signed key, the server should verify that the Signature Key, as specified by the client inside the Key
Wrapper Data, has the "Verify" bit set in the Cryptographic Usage Mask. If the wrapping key is not
permitted to be used for the requested purpose (e.g., when the Cryptographic Usage Mask is not set), the

server should return the Operation Failed error.

3.21.1 Encrypt-only Example with a Symmetric Key as an Encryption Key for a Get Request and Response

The client sends a Get request to obtain a key that is stored on the server. When the client sends a Get request to the server, a Key Wrapping Specification may be included. If a Key Wrapping Specification is included in the Get request, and a client wants the requested key and its Cryptographic Usage Mask
attribute to be wrapped <u>withusing</u> AES key wrap, the client includes the following information in the Key Wrapping Specification:

Wrapping Method: Encrypt

692 693

694

697

- Encryption Key Information
 - Unique Key ID: Key ID of the AES wrapping key
- 695 Cryptographic Parameters: The Block Cipher Mode is NISTKeyWrap (not necessary if default 696 block cipher mode for wrapping key is NISTKeyWrap)
 - Attribute Name: Cryptographic Usage Mask

The server uses the Unique Key ID specified by the client to determine the attributes set for the proposed
 wrapping key. For example, the algorithm of the wrapping key is not explicitly specified inside the Key
 Wrapping Specification. The ; the server determines the algorithm to be used for wrapping the key by

identifying the Algorithm attribute set for the specified Encryption Key.

The Cryptographic Parameters attribute should be specified by the client if multiple instances of the

Cryptographic Parameters exist, and the lowest index does not correspond to the NIST key wrap mode of operation. The server should verify that the AES wrapping key has NISTKeyWrap set as an allowable

705 Block Cipher Mode, and that the "Wrap Key" bit is set in the Cryptographic Usage Mask.

106 If the correct data was provided to the server, and no conflicts exist, the server <u>AES key</u> wraps the Key 107 Value (both the Key Material and the Cryptographic Usage Mask attribute) for the requested key

withusing the AES key wrap algorithm and wrapping key specified in the Encryption Key Information. The

kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved. 18 March 201005 Page 23 of 46

- 709 wrapped key; the Key Value contains both the Key Material and the Cryptographic Usage Mask attribute,
 710 and return the encrypted result (byte string) is returned as the Key Value in the Key Block of the server's
 711 response inside the Key Value of the Key Block.
- 712 The Key Wrapping Data of the Key Block in the Get Response Payload includes the same data as
- 713 specified in the Key Wrapping Specification of the Get Request Payload except for the Attribute Name.

3.21.2 Encrypt-only Example with a Symmetric Key as an Encryption Key for a Register Request and Response

- 716 The client sends a Register request to the server and includes the wrapped key and the <u>Uniqueunique</u> ID of the wrapping key inside the Request Payload. The wrapped key is provided to the server inside the Key Block. The Key Block includes the Key Value Type, the Key Value, and the Key Wrapping Data. The Key Value Type identifies the format of the Key Material, the Key Value consists of the Key Material and optional attributes that may be included to cryptographically bind the attributes to the Key Material, and
- the Key Wrapping Data identifies the wrapping mechanism and the encryption key used to wrap the
- 722 object and the wrapping mechanism.
- Similar to the example in 3.21.1_the key is wrapped using the AES key wrap. The Key Value includes four
 attributes: Cryptographic Algorithm, Cryptographic Length, Cryptographic Parameters, and Cryptographic
 Usage Mask.
- 726 The Key Wrapping Data includes the following information:
- Wrapping Method: Encrypt

728 729

730

731

- Encryption Key Information
 - Unique Key ID: Key ID of the AES wrapping key
- Cryptographic Parameters: The Block Cipher Mode is NISTKeyWrap (not necessary if default block cipher mode for wrapping key is NISTKeyWrap)

Attributes do not need to be specified in the Key Wrapping Data. When registering a wrapped Key Value
 with attributes, clients may include these attributes inside the Key Value without specifying them inside
 the Template-Attribute.

- Prior to unwrapping the key, the server determines the wrapping algorithm from the Algorithm attribute set for the specified Unique ID in the Encryption Key Information. The server verifies that the wrapping key may be used for the specified purpose. In particular, if the client includes the Cryptographic Parameters in the Encryption Key Information, the server verifies that the specified Block Cipher Mode is set for the wrapping key. The server also verifies that the wrapping key has the "Unwrap Key" bit set in the
- 740 Cryptographic Usage Mask.

The Register Response Payload includes the Unique ID of the newly registered key and an optional list of attributes that were implicitly set by the server.

3.21.3 Encrypt-only Example with an Asymmetric Key as an Encryption Key for a Get Request and Response

The client sends a Get request to obtain a key (either symmetric or asymmetric) that is stored on the
server. When the client sends a Get request to the server, a Key Wrapping Specification may be
included. If a Key Wrapping Specification is included, and the key is to be wrapped with an RSA public
key using the OAEP encryption scheme, the client includes the following information in the Key Wrapping
Specification. Note that for this example, attributes for the requested key are not requested.

- Wrapping Method: Encrypt
- Encryption Key Information
- 752 Unique Key ID: Key ID of the RSA public key
- 753 Cryptographic Parameters:
- 754 Padding Method: OAEP
- 755 Hashing Algorithm: SHA-256

kmip-ug-1.0-cd-09

Copyright © OASIS® 2010.2009. All Rights Reserved.

<u>18 March 2010</u> Page 24 of 46

- 756 The Cryptographic Parameters attribute is specified by the client if multiple instances of Cryptographic
- 757 Parameters exist for the wrapping key, and the lowest index does not correspond to the associated
- 758 padding method. The server should verify that the specified Cryptographic Parameters in the Key
- 759 Wrapping Specification and the "Wrap Key" bit in the Cryptographic Usage Mask are set for the 760 corresponding wrapping key.
- 761 The Key Wrapping Data returned by the server in the Key Block of the Get Response Payload includes 762 the same data as specified in the Key Wrapping Specification of the Get Request Payload.
- For both OAEP and PSS, KMIP currently assumes that the Hashing Algorithm specified in the 763
- 764 Cryptographic Parameters of the Get request is used for both the Mask Generation Function (MGF) and
- hashing data. The example above requires the server to use SHA-256 for both purposes. 765

3.21.4 MAC-only Example with an HMAC Key as an Authentication Key for 766 a Get Request and Response 767

768 The client sends a Get request to obtain a key that is stored on the server. When the client sends a Get 769 request to the server, a Key Wrapping Specification may be included. If a key and Custom Attribute (i.e., x-Nonce) is to be MACed with HMAC SHA-256, the following Key Wrapping Specification is specified: 770

771 Wrapping Method: MAC/sign

772

773 774

784

785

786

- MAC/Signature Key Information .
 - Unique Key ID: Key ID of the MACing key (note that the algorithm associated with this key would be HMAC-256)
- 775 Attribute Name: x-Nonce

776 For HMAC, no Cryptographic Parameters need to be specified, since the algorithm, including the hash 777 function, may be determined from the Algorithm attribute set for the specified MAC Key. The server should verify that the HMAC key has the "MAC Generate" bit set in the Cryptographic Usage Mask. Note 778 779 that an HMAC key does not require the "Wrap Key" bit to be set in the Cryptographic Usage Mask.

780 The server creates an HMAC value over the Key Value if the specified MACing key may be used for the specified purpose and no conflicts exist. The Key Value is returned in plaintext, and the Key Block 781 includes the following Key Wrapping Data: 782

- 783 Wrapping Method: MAC/sign
 - MAC/Signature Key Information •
 - Unique Key ID: Key ID of the MACing key
 - MAC/Signature: HMAC result of the Key Value

787 In the example, the custom attribute x-Nonce was included to help clients, who are relying on the proxy model, to detect replay attacks. End-clients, who communicate with the key management server, may not 788 789 support SSL/TLS and may not be able to rely on the message protection mechanisms provided by a security protocol. An alternative approach for these clients would be to use the custom attribute may be 790 791 created to hold a random number, counter, nonce, date, or time. The custom attribute needs to be 792 created before requesting the server to return a wrapped key and is recommended to be set if clients

793 frequently wrap/sign the same key with the same wrapping/signing key.

3.21.5 Registering a Wrapped Key as an Opaque Cryptographic Object 794

795 Clients may want to register and store a wrapped key on the server without the server being able to 796 unwrap the key (i.e., the wrapping key is not known to the server). Instead of storing the wrapped key as 797 an opaque object, clients have the option to store the wrapped key inside the Key Block as an opaque 798 cryptographic object, i.e., the wrapped key is registered as a managed cryptographic object, but the 799 encoding of the key is unknown to the server. Registering an opaque cryptographic object allows clients to set all the applicable attributes that apply to cryptographic objects (e.g., Cryptographic Algorithm and

- 800 801 Cryptographic Length),
- 802 Opaque cryptographic objects are set by specifying the following inside the Key Block structure:

kmip-ua-1.0-cd-09 Copyright © OASIS® 2010, 2009. All Rights Reserved. 18 March 201005 Page 25 of 46

- 803 Key Format Type: Opaque
- Key Material: Wrapped key as a Byte String
- 805 The Key Wrapping Data does not need to be specified.

806 3.22 Object Group

807 The key management system may specify rules for valid group names which may be created by the 808 client. Clients are informed of such rules by a mechanism that is not specified by [KMIP-Spec]. In the protocol, the group names themselves are character strings of no specified format. Specific key 809 810 management system implementations may choose to support hierarchical naming schemes or other 811 syntax restrictions on the names. Groups may be used to associate objects for a variety of purposes. A 812 set of keys used for a common purpose, but for different time intervals, may be linked by a common 813 Object Group. Servers may create predefined groups and add objects to them independently of client 814 requests.

815 3.23 Certify and Re-certify

The key management system may contain multiple embedded CAs or may have access to multiple external CAs. How the server routes a certificate request to a CA is vendor-specific and outside the scope of KMIP. If the server requires and supports the capability for clients to specify the CA to be used for signing a Certificate Request, then this information may be provided by including the Certificate Issuer attribute in the Certify or Re-certify request.

821 [KMIP-Spec] supports multiple options for submitting a certificate request to the key management server 822 within a Certify or Re-Certify operation. It is a vendor decision as to whether the key management server 823 offers certification authority (CA) functionality or proxies the certificate request onto a separate CA for 824 processing. The type of certificate request formats supported is also a vendor decision, and this may, in 825 part, be based upon the request formats supported by any CA to which the server proxies the certificate 826 requests.

All certificate request formats for requesting X.509 certificates specified in [KMIP-Spec] (i.e., PKCS#10, PEM and CRMF) provide a means for allowing the CA to verify that the client that created the certificate request possesses the private key corresponding to the public key in the certificate request. This is
referred to as Proof-of-Possession (POP). However, it should be noted that in the case of the CRMF format, some CAs may not support the CRMF POP option, but instead rely upon the underlying certificate management protocols (i.e., CMP and CMC) to provide POP. In the case where the CA does not support POP via the CRMF format (including CA functionality within the key management server), an alternative

certificate request format (i.e., PKCS#10, PEM) would need to be used if POP needs to be verified.

835 3.24 Specifying Attributes during a Create Key Pair Operation

836 The Create Key Pair operation allows clients to specify attributes using the Common Template-Attribute,

Private Key Template-Attribute, and Public Key Template-Attribute. The Common Template-Attribute object includes a list of attributes that apply to both the public and private key. Attributes that are not

common to both keys may be specified using the Private Key Template-Attribute or Public Key Template-

Attribute. If a single-instance attribute is specified in multiple Template-Attribute objects, the server obeys the following order of precedence:

- 842 1. Attributes specified explicitly in the Private and Public Key Template-Attribute, then
- 843 2. Attributes specified via templates in the Private and Public Key Template-Attribute, then
- 3. Attributes specified explicitly in the Common Template-Attribute, then
- 845 4. Attributes specified via templates in the Common Template-Attribute

3.24.1 Example of Specifying Attributes during the Create Key Pair Operation

A client specifies several attributes in the Create Key Pair Request Payload. The Common Template-848 849 Attribute includes the template name RSACom and other explicitly specified common attributes: 850 Common Template-Attribute 851 RSACom Template 852 Cryptographic Algorithm: RSA _ 853 Cryptographic Length: 2048 _ 854 Cryptographic Parameters: Padding Method OAEP 855 Custom Attribute: x-Serial 1234 _ Object Group: Key encryption group 1 856 _ 857 Attribute Cryptographic Length: 4096 858 _ Cryptographic Parameters: Padding Method PKCS1 v1.5 859 _ 860 Custom Attribute: x-ID 56789 861 The Private Key Template-Attribute includes the template name RSAPriv and other explicitly-specified 862 private key attributes: 863 Private Key Template-Attribute 864 **RSAPriv** Template • 865 - Object Group: Key encryption group 2 866 Attribute 867 Cryptographic Usage Mask: Unwrap Key _ Name: PrivateKey1 868 _ The Public Key Template Attribute includes explicitly-specified public key attributes: 869 870 Public Key Template-Attribute 871 Attribute 872 Cryptographic Usage Mask: Wrap Key 873 Name: PublicKey1 _ 874 875 Following the attribute precedence rule, the server creates a 4096-bit RSA key. The following client-876 specified attributes are set: 877 Private Key 878 Cryptographic Algorithm: RSA • 879 Cryptographic Length: 4096 ٠ 880 Cryptographic Parameters: OAEP ٠ 881 Cryptographic Parameters: PKCS1 v1.5 • 882 Cryptographic Usage Mask: Unwrap Key ٠ 883 Custom Attribute: x-Serial 1234 ٠ Custom Attribute: x-ID 56789 884 • 885 Object Group: Key encryption group 1 • 886 Object Group: Key encryption group 2 . 887 Name: PrivateKey1 • 888 Public Key kmip-ug-1.0-cd-09 18 March 201005

Copyright © OASIS® 2010.2009. All Rights Reserved.

Page 27 of 46

- 889 Cryptographic Algorithm: RSA
- 890 Cryptographic Length: 4096
- 891 Cryptographic Parameters: OAEP
- Cryptographic Parameters: PKCS1 v1.5 892
- Cryptographic Usage Mask: Wrap Key 893 .
- 894 Custom Attribute: x-Serial 1234 •
- 895 Custom Attribute: x-ID 56789
 - Object Group: Key encryption group 1
- 897 Name: PublicKey1

896

3.25 Registering a Key Pair 898

899 During a Create Key Pair operation, a Link Attribute is automatically created by the server for each object 900 (i.e., a link is created from the private key to the public key and vice versa). Certain attributes are the 901 same for both objects and are set by the server while creating the key pair. The KMIP protocol does not 902 support an equivalent operation for registering a key pair. Clients are able to register the objects 903 independently and manually set the Link attributes to make the server aware that these keys are 904 associated with each other. When the Link attribute is set for both objects, the server should verify that 905 the registered objects indeed correspond to each other and apply similar restrictions as if the key pair was 906 created on the server. 907 Clients should perform the following steps when registering a key pair: 908 1. Register the public key and set all associated attributes: 909 a. Cryptographic Algorithm 910 b. Cryptographic Length c. Cryptographic Usage Mask 911 912 2. Register the private key and set all associated attributes 913 a. Cryptographic Algorithm is the same for both public and private key 914 b. Cryptographic Length is the same for both public and private key 915 Cryptographic Parameters may be set; if set, the value is the same for both the public and C. 916 private key 917 d. Cryptographic Usage Mask is set, but does not contain the same value for both the public 918 and private key 919 e. Link is set with Link Type Public Key Link and the Linked Object Identifier of the 920 corresponding Public Key 921 f. Link is set for the Public Key with Link Type Private Key Link and the Linked Object Identifier 922 of the corresponding Private Key 923 3.26 Non-Cryptographic Objects 924 The KMIP protocol allows clients to register Secret Data objects. Secret Data objects may include

- 925 passwords or data that are used to derive keys.
- KMIP defines Secret Data as cryptographic objects. Even if the object is not used for cryptographic 926 927 purposes, clients still set certain attributes, such as the Cryptographic Usage Mask, for this object unless 928 otherwise stated. Similarly, servers set certain attributes for this object, including the Digest, State, and certain Date attributes, even if the attributes seem relevant only for cryptographic objects. 929
- 930 When registering a Secret Data object, the following attributes are set by the server:

kmip-ua-1.0-cd-09

Copyright © OASIS® 2010.2009. All Rights Reserved.

- Unique Identifier
- Object Type
- 933 Digest
- 934 State

939

- 935 Initial Date
- 936 Last Change Date

937 When registering a Secret Data object for non-cryptographic purposes, the following attributes are set by 938 either the client or the server:

Cryptographic Usage Mask

940 3.27 Asymmetric Concepts with Symmetric Keys

The Cryptographic Usage Mask attribute is intended to adequately support asymmetric concepts using
symmetric keys. This is fairly common practice in established crypto systems: the MAC is an example of
an operation where a single symmetric key is used at both ends, but policy dictates that one end may
only generate cryptographic tokens using this key (the MAC) and the other end may only verify tokens.
The security of the system fails if the verifying end is able to use the key to perform generate operations.

946 In these cases it is not sufficient to describe the usage policy on the keys in terms of cryptographic 947 primitives like "encrypt" vs. "decrypt" or "sign" vs. "verify". There are two reasons why this is the case.

- In some of these operations, such as MAC generate and verify, the same cryptographic primitive is used in both of the complementary operations. MAC generation involves computing and returning the MAC, while MAC verification involves computing that same MAC and comparing it to a supplied value to determine if they are the same. Thus, both generation and verification use the "encrypt" operation, and the two usages are not able to be distinguished by considering only "encrypt" vs. "decrypt".
- Some operations which require separate key types use the same fundamental cryptographic
 primitives. For example, encryption of data, encryption of a key, and computation of a MAC all
 use the fundamental operation "encrypt", but in many applications, securely differentiated keys
 are used for these three operations. Simply looking for an attribute that permits "encrypt" is not
 sufficient.

Allowing the use of these keys outside of their specialized purposes may compromise security. Instead,
 specialized application-level permissions are necessary to control the use of these keys. KMIP provides
 several pairs of such permissions in the Cryptographic Usage Mask (3.14), such as:

MAC GENERATE MAC VERIFY	For cryptographic MAC operations. Although it is possible to compose certain MACs using a series of encrypt calls, the security of the MAC relies on the operation being atomic and specific.
GENERATE CRYPTOGRAM VALIDATE CRYPTOGRAM	For composite cryptogram operations such as financial CVC or ARQC To specify exactly which cryptogram the key is used for it is also necessary to specify a <i>role</i> for the key (see Section 3.6 "Cryptographic Parameters" in [KMIP-Spec]).
TRANSLATE ENCRYPT TRANSLATE DECRYPT TRANSLATE WRAP TRANSLATE UNWRAP	To accommodate secure routing of traffic and data. In many areas that rely on symmetric techniques (notably, but not exclusively financial networks), information is sent from place to place encrypted using shared symmetric keys. When encryption keys are changed, it is desirable for the change to be an atomic operation, otherwise distinct unwrap- wrap or decrypt-encrypt steps risk leaking the

kmip-ug-1.0-cd-09

Copyright © OASIS® 2010.2009. All Rights Reserved.

plaintext data during the translation process.
TRANSLATE ENCRYPT/DECRYPT is used for data encipherment.
TRANSLATE WRAP/UNWRAP is used for key wrapping.

962 Table 2: Cryptographic Usage Masks Pairs

In order to support asymmetric concepts using symmetric keys in a KMIP system, the server
implementation needs to be able to differentiate between clients for generate operations and clients for
verify operations. As indicated by Section 3 ("Attributes") of [KMIP-Spec] there is a single key object in
the system to which all relevant clients refer, but when a client requests that key, the server is able to
choose which attributes (permissions) to send with it, based on the identity and configured access rights
of that specific client. There is, thus, no need to maintain and synchronize distinct copies of the symmetric
key – just a need to define access policy for each client or group of clients.

The internal implementation of this feature at the server end is a matter of choice for the vendor: storing multiple key blocks with all necessary combinations of attributes or generating key blocks dynamically are

972 both acceptable approaches.

973 3.28 Application Specific Information

974 The Application Specific Information attribute is used to store data which is specific to the application(s) 975 using the object. Some examples of Application Name Space and Application Data pairs are given below.

- 976 SMIME, 'someuser@company.com'
- 977 <u>TLSSSL</u>, 'some.domain.name'
- 978 Volume Identification, '123343434'
 - File Name, 'secret.doc'
- Client Generated Key ID, '450994003'

981 The following Application Name Spaces are recommended:

982 • SMIME

979

988

989

990

998

- 983 TLS
- 984 SSL
- 985 IPSEC
- 986 HTTPS
- 987 PGP
 - Volume Identification
 - File Name
 - LTO4
- 991 LIBRARY-LTO4

KMIP provides optional support for server-generated Application Data. Clients may request the server to
 generate the Application Data for the client by omitting Application Data while setting or modifying the
 Application Specific Information attribute. A server only generates the Application Data if the Application
 Data is completely omitted from the request, and the client-specified Application Name Space is

996 recognized and supported by the server. An example for requesting the server to generate the Application 997 Data is shown below:

- AddAttribute(UID, AppSpecInfo{AppNameSpace='LIBRARY-LTO4'});
- 999 If the server does not recognize the name space, the "Application Name Space Not Supported" error is 1000 returned to the client.

kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved. 18 March 201005 Page 30 of 46

- 1001 If the Application Data is set to null, as shown in the example below, and the Application Name Space is 1002 recognized by the server, the server does not generate the Application Data for the client. The server
- stores the Application Specific Information attribute with the Application Data value set to null.
 - AddAttribute(UID, AppSpecInfo{AppNameSpace='LIBRARY-LTO4', AppData=null});

1005 3.29 Mutating Attributes

1004

- KMIP does not support server mutation of client-supplied attributes. If a server does not accept an
 attribute value that is being specified inside the request by the client, the server returns an error and
 specifies "Invalid Field" as Result Reason.
- Attributes that are not set by the client, but are implicitly set by the server as a result of the operation, may optionally be returned by the server in the operation response inside the Template–Attribute.
- 1011 If a client sets a time-related attribute to the current date and time (as perceived by the client), but as a
- 1012 result of a clock skew, the specified date of the attribute is earlier than the time perceived by the server,
- 1013 the server's policy will be used to determine whether to accept the "backdated attribute". KMIP does not 1014 require the server to fail a request if a backdated attribute is set by the client.
- 1015 If a server does not support backdated attributes, and cryptographic objects are expected to change state
- 1016 at the specified current date and time (as perceived by the client), clients are recommended to issue the
- 1017 operation that would implicitly set the date for the client. For example, instead of explicitly setting the
- 1018 Activation Date, clients could issue the Activate operation. This would require the server to set the
- 1019 Activation Date to the current date and time as perceived by the server.
- 1020 If it is not possible to set a date attribute via an operation, and the server does not support backdated 1021 attributes, clients need to take into account that potential clock skew issues may cause the server to
- 1022 return an error even if a date attribute is set to the client's current date and time.
- 1023
 For additional information, refer to the sections describing the State attribute and the Time Stamp field in

 1024
 [KMIP-Spec].

1025 3.30 Interoperable Key Naming for Tape

1026 This section describes methods for creating and storing key identifiers that are interoperable across multi-1027 vendor KMIP clients.

1028 3.30.1 Native Tape Encryption by a KMIP Client

- 1029 This method is primarily intended to promote interoperable key naming between tape library products 1030 which already support non-KMIP key managers, where KMIP support is being added.
- 1031 When those existing library products become KMIP clients, a common method for naming and storing
- 1032 keys may be used to support moving tape cartridges between the libraries, and successfully retrieving 1033 keys, assuming that the clients have appropriate access privileges. The library clients may be from
- 1034 multiple vendors, and may be served by a KMIP key manager from a different vendor.

1035 3.30.1.1 Method Overview

- The method uses the KMIP Application Specific Information (ASI) attribute's Application Data field to store the key name. The ASI Application Name Space is used to identify the namespace (such as LIBRARY-LTO4).
- The method also uses the tape format's Key Associated Data (KAD) fields to store the key name.
 Tape formats may provide both authenticated and unauthenticated storage for the KAD data. This method ensures optimum utilization of the authenticated KAD data when the tape format supports authentication.
- The method supports both client-generated and server-generated key names.
- 1044• The method, in many cases, is backward-compatible if tapes are returned to a non-KMIP key
manager environment.

046	٠	Key names stored in the KMIP server's ASI attribute are always text format. Key names stored on
047		the KMIP client's KAD fields are always numeric format, due to space limitations of the tape
048		format. The method basically consists of implementing a specific algorithm for converting
049		between text and numeric formats.

• The algorithm used by this conversion is reversible.

1051 3.30.1.2 Definitions

- Key Associated Data (KAD). Part of the tape format. May be segmented into authenticated and unauthenticated fields. KAD usage is detailed in the SCSI SSC-3 standard from the T10 organization.
- 1055 Application Specific Information (ASI). A KMIP attribute.
- Hexadecimal numeric characters. Case-sensitive, printable, single byte ASCII characters representing the numbers 0 through 9 and uppercase alpha A through F. (US-ASCII characters 30h-39h and 41h-46h).
- Hexadecimal numeric characters are always paired, each pair representing a single 8-bit numeric
 value. A leading zero character is provided, if necessary, so that every byte in the tape's KAD is
 represented by exactly 2 hexadecimal numeric characters.
- N(k). The number of bytes in the tape format's combined KAD fields (both authenticated and unauthenticated).
- N(a), N(u). The number of bytes in the tape format's authenticated, and unauthenticated KAD fields, respectively.

1067 3.30.1.3 Algorithm 1. Numeric to text direction (tape format's KAD to KMIP 1068 ASI)

Description: All information contained in the tape format's KAD fields is converted to a null-terminated
 ASCII string consisting of hexadecimal numeric character pairs. First, the unauthenticated KAD data is
 converted to text. Then, the authenticated KAD data is converted and appended to the end of the string.
 The string is then null-terminated.

1074 Implementation Example:

1073

1088

- Define an input buffer sized for N(k). For LTO4, N(k) is 44 bytes (12 bytes authenticated, 32 unauthenticated).
- 1077 2. Define an output buffer sufficient to contain a null-terminated string with a maximum length of $2^*N(k)+1$ bytes.
- Define the standard POSIX (also known as C) locale. Each character in the string is a single-byte US-ASCII character.
- Copy the tape format's KAD data, from the unauthenticated KAD field first, to the input buffer.
 Effectively, the first byte (byte 0) of the input buffer is the first byte of unauthenticated KAD. Bytes
 from the authenticated KAD are concatenated, after the unauthenticated bytes.
- 1084 5. For each byte in the input buffer, convert to US-ASCII as follows:
- a. Convert the byte's value to exactly 2 hexadecimal numeric characters, including a leading 0
 where necessary. Append these 2 numeric characters to the output buffer, with the high-nibble
 represented by the left-most hexadecimal numeric character.
 - b. After all byte values have been converted, null terminate the output buffer.
- When storing the string to the KMIP server, use the object's ASI attribute's Application Data field.
 Store the namespace (such as LIBRARY-LTO4) in the ASI attribute's Application Name Space field.

kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved. 18 March 201005 Page 32 of 46

1091**3.30.1.4**Algorithm 2. Text to numeric direction (KMIP ASI to tape format's1092KAD)

1093 Description: Hexadecimal numeric character pairs in the null-terminated ASCII string are converted to 1094 single byte numeric values, and stored in the tape format's KAD fields. The authenticated KAD field is 1095 populated first, from a sub-string consisting of the <u>last</u> 2*N(a) characters in the full string. Any remaining 1096 characters in the string are converted and stored to the unauthenticated KAD field. The null termination 1097 byte is not converted.

1099 Implementation Example:

- Obtain the key's name from the KMIP server's ASI attribute for that object. Copy the null terminated string to an input buffer of size 2*N(k) + 1 bytes. For LTO4, an 89 character string, including null termination, is sufficient for all possible key descriptors when names are directly referenced.
- Define output buffers for unauthenticated KAD, and authenticated KAD, of size N(u) and N(a)
 respectively. For LTO4, this would be 32 bytes of unauthenticated data, and 12 bytes of authenticated data.
- Define the standard POSIX (also known as C) locale. Each character in the string is a single-byte US-ASCII character.
- First, populate the authenticated KAD buffer, converting a sub-string consisting of the <u>last</u> 2*N(a)
 characters of the full string, not including the null termination byte.
- 1110 5. When the authenticated KAD is filled, next populate the unauthenticated KAD buffer, by converting
 1111 the remaining hexadecimal character pairs in the string.

1112 3.30.1.5 Example Output

1113 The following are examples illustrating some results of this method. In the following examples, the sizes 1114 of the KAD for LTO4 are used. Different tape formats may utilize different KAD sizes.

1115

1098

- 1116 Example 1. Full combined KAD
- 1117
- This LTO4 tape's combined KAD contains the following data (represented in hexadecimal). For LTO4, the
 unauthenticated KAD contains 32 bytes, and the authenticated KAD contains 12 bytes.
- 1121 Example 1a. Hexadecimal numeric data from a tape's KAD.
- 1122 Shaded data is authenticated by the tape drive.

1123					
1124	02 04 17 11	39 43 42 36	30 41 3	33 34 39	31 44 33

1125	41 41 43 36 32 42 07 F6 54 54 32 36 30 38 4C 34
1120	

1126 30 30 30 39 30 35 32 38 30 34 31 32

1127

The algorithm converts the numeric KAD data to the following 89 character null-terminated string for
 storage in the Application Data field of a KMIP object's Application Specific Information attribute. The ASI
 Application Name Space contains "LIBRARY-LTO4".

- 1131
- 1132 Example 1b. Text string from KMIP ASI Application Data.1133 Shaded characters are derived from authenticated data. The null character is represented as
- 1134 <null>
- 1135

kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved. <u>18 March 2010</u> Page 33 of 46

1136 1137 1138	0204171139434236304133343931443341414336324207F65454323630384C34 <mark>3030303930353</mark> 23830343132 <null></null>	
1139 1140 1141	Example 1c. The hexadecimal values of the 89 US-ASCII characters in string 1b, from the KMIP ASI Application Data. Note: these values are always in the range 30h-39h, or in the range 41h-46h, or the 0h null.	
1142 1143 1144	30 32 30 34 31 37 31 31 33 39 34 33 34 32 33 36 33 30 34 31 33 33 33 34 33 39 33 31 34 34 33 33 34 31 34 34 33 33 34 31 34 34 33 33 36 33 32 34 32 30 37 46 36 35 34 35 34 33 32 33 36 33 30 33 38 34 43 33 34 33 30 33 30 33 30 33 39 33 30 33 35 33 32 33 38 33 30 33 34 33 31 33 32 00	
1145 1146 1147 1148 1149	For the reverse transformation, a client would retrieve the string in 1b from the server, derive the numeric values shown in 1a, and store them to the tape format's KAD data. First, the sub-string containing the right-most 24 characters of the full 1b string are used to derive the 12-byte authenticated KAD. The remaining characters are used to derive the 32-byte unauthenticated KAD.	
1150		
1151	Example 2. Authenticated KAD only	
1152 1153	This LTO4 tape's KAD contains the following data (represented in hexadecimal), all 12 bytes obtained from the authenticated KAD field. There is no unauthenticated KAD data.	
1154		
1155	Example 2a. Hexadecimal numeric data from a tape's KAD.	
1156	Shaded data is authenticated.	
1158	17 48 33 C6 20 42 10 A7 E8 05 F8 C7	
1159 1160	The algorithm converts the numeric KAD data to the following 24 character null-terminated string, for storage in the Application Data field of a KMIP object's Application Specific Information attribute.	
1161		
1162	Example 2b. Text string from KMIP ASI Application Data.	
1163 1164 1165	Shaded characters are derived from authenticated data. The null character is represented as <null></null>	
1166	174833C6204210A7E805F8C7 <null></null>	
1167		
1168 1169 1170	For the reverse transformation, a client would derive the numeric values in 2a, and store them to the tape format's KAD data. The right-most 24 characters of the string in 2b are used to derive the 12 byte authenticated KAD. In this example, there is no unauthenticated KAD data.	
1171		
1172	Example 3. Partially filled authenticated KAD originating from a non-KMIP method	
1173 1174	This LTO4 tape's KAD contains the following data (represented in hexadecimal). The unauthenticated KAD contains 10 bytes, and the authenticated KAD contains 8 bytes.	
1175		
1176 1177 1178	Since the authenticated KAD was not filled, but the unauthenticated data was populated, the method creating this key name is potentially not backward-compatible with the KMIP key naming method. See backward-compatibility assessment, below.	
1179		
1180	Example 3a. Hexadecimal numeric data from a non-KMIP tape's KAD.	
1181 1182	Shaded data is authenticated.	
1183	02 04 17 11 39 43 42 36 30 41 30 30 30 39 30 35	
	kmip-ug-1.0-cd- <u>09 18 March 2010</u> 95	5 November 2009
l	Copyright © OASIS® 2010,2009. All Rights Reserved. Page 34 of 46	

1184	32 38
1185	
1186 1187	The algorithm converts the numeric KAD data to the following 36 character null-terminated string, for storage in the Application Data field of a KMIP object's Application Specific Information attribute.
1188	
1189	Example 3b. Text string from KMIP ASI Application Data.
1190 1191	Shaded characters are derived from authenticated data. The null character is represented as <null></null>
1192	
1193	02041711394342363041 <mark>3030303930353238</mark> <null></null>
1194	
1195 1196 1197 1198	For the reverse transformation, a client would derive the same numeric values shown in 3a, and store them to the tape's KAD. But their storage locations within the KAD now differs (see 3c). The right-most 24 characters from the text string in 3b are used to derive the 12-byte authenticated KAD. The remaining characters are used to fill the 32-byte unauthenticated KAD.
1199	
1200	Example 3c. Hexadecimal numeric data from a tape's KAD.
1201 1202	Shaded data is authenticated.
1203	02 04 17 11 39 43 42 36 30 41 30 30 30 39 30 35
1204	32 38
1205	3.30.1.6 Backward-compatibility assessment
1206 1207	Where all the following conditions exist, a non-KMIP solution may encounter compatibility issues during the Read and Appended Write use cases.
1208 1209	 The tape format supports authenticated KAD, but the non-KMIP solution does not use, or only partially uses, the authenticated KAD field.
1210	2. The non-KMIP solution is sensitive to data position within the combined KAD.
1211 1212	3. The media was written in a KMIP environment, using this method, then moved to the non-KMIP environment
1213	3.31 Revocation Reason Codes
1214 1215 1216 1217 1218 1219 1220 1221 1222	The enumerations for the Revocation Reason attribute specified in KMIP (see table 9.1.3.2.17 in [KMIP- Spec]) are aligned with the Reason Code specified in X.509 and referenced in RFC 5280 with the following exceptions. The <i>certificateHold</i> and <i>removeFromCRL</i> reason codes have been excluded from [KMIP-Spec], since this version of KMIP does not support certificate suspension (putting a certificate hold) or unsuspension (removing a certificate from hold). The <i>aaCompromise</i> reason code has been excluded from [KMIP-Spec] since it only applies to attribute certificates, and, at this point of time, attribute certificates are considered out-of-scope for [KMIP-Spec]. The <i>priviledgeWithdrawn</i> reason code is included in [KMIP-Spec] since it may be used for either attribute or public key certificates. In the context of its use within KMIP it is assumed to only apply to public key certificates.
1223	3.32 Certificate Renewal, Update, and Re-key
1224 1225 1226	The process of generating a new certificate to replace an existing certificate may be referred to by multiple terms, based upon what data within the certificate is changed when the new certificate is created. In all situations, the new certificate includes a new serial number and new validity dates[KMIP-Spec]

1226In all situations, the new certificate includes a new serial number and new validity dates. -
uses the following terminology which is aligned with the definitions found in IETF RFCs [RFC3647]36471228and [RFC4949]4949:

kmip-ug-1.0-cd-<u>09</u> Copyright © OASIS® <u>2010,</u>2009. All Rights Reserved. 18 March 201005 Page 35 of 46

- Certificate Renewal: The issuance of a new certificate to the subject without changing the subject public key or other information (except the serial number and certificate validity dates) in the certificate.
- Certificate Update: The issuance of a new certificate, due to changes in the information in the certificate other than the subject public key.
- Certificate Rekey: The generation of a new key pair for the subject and the issuance of a new certificate that certifies the new public key.

1236 The current KMIP Specification supports certificate renewals using the Re-Certify operation and certificate 1237 updates using the Certify operation. Support for certificate rekey is not currently supported by KMIP, since 1238 certificate rekey requires the ability to rekey an asymmetric key pair a capability not currently supported 1239 by KMIP. Support for rekey of asymmetric key pairs, along with certificate rekey, may be considered for a 1240 future KMIP release.

1241 3.33 Key Encoding

1242Two parties receiving the same key as a Key BYTE STRING make use of the key in exactly the same1243way in order to interoperate. To ensure that, it is necessary to define a correspondence between the1244abstract syntax of Key and the notation in the standard algorithm description that defines how the key is1245used. The next sections establish that correspondence for the algorithms AES [FIPS197] and Triple-DES1246[SP800-67][SP800-67].

1247 3.33.1 AES Key Encoding

1248[FIPS197] section 5.2, titled Key Expansion, uses the input key as an array of bytes indexed starting at 0.1249The first byte of the Key becomes the key byte in AES that is labeled index 0 in [FIPS197] and the other1250key bytes follow in index order.

Proper parsing and key load of the contents of the Key for AES is determined by using the following Key byte string to generate and match the key expansion test vectors in [FIPS197] Appendix A for the 128-bit (16 byte) AES Cipher Key: 2B 7E 15 16 28 AE D2 A6 AB F7 15 88 09 CF 4F 3C.

1254 3.33.2 Triple-DES Key Encoding

1255 A Triple-DES key consists of three keys for the cryptographic engine (Key1, Key2, and Key3) that are

each 64 bits (even though only 56 are used); the three keys are also referred to as a key bundle (KEY)
 [SP800-67][SP800-67]. A key bundle may employ either two or three mutually independent keys. When only two are employed (called two-key Triple-DES), then Key1 = Key3.

1259 Each key in a Triple-DES key bundle is expanded into a key schedule according to a procedure defined in

[SP800,67][SP800-67] Appendix A. That procedure numbers the bits in the key from 1 to 64, with
 number 1 being the left-most, or most significant bit. The first byte of the Key is bits 1 through 8 of Key1,
 with bit 1 being the most significant bit. The second byte of the Key is bits 9 through 16 of Key1, and so
 forth, so that the last byte of the KEY is bits 57 through 64 of Key3 (or Key2 for two-key Triple-DES).

1264 Proper parsing and key load of the contents of Key for Triple-DES is determined by using the following

1264 Froper parsing and key load of the contents of Key for Triple-DE's is determined by doing the following
 1265 Key byte string to generate and match the key expansion test vectors in [SP800-67][SP800-67] Appendix
 1266 B for the key bundle:

- 1267 Key1 = 0123456789ABCDEF
- 1268 Key2 = 23456789ABCDEF01
- 1269 Key3 = 456789ABCDEF0123

1270 3.34 Using the Same Asymmetric Key Pair in Multiple Algorithms

1271There are mathematical relationships between certain asymmetric cryptographic algorithms such as the
Digital Signature Algorithm (DSA) and Diffie-Hellman (DH) and their elliptic curve equivalents ECDSA and
ECDH that allow the same asymmetric key pair to be used in both algorithms. In addition, one will notice

1274 overlaps in the key format used to represent the asymmetric key pair for each algorithm type.

kmip-ug-1.0-cd-09 Copyright © OASIS® 2010.2009. All Rights Reserved. 18 March 201005 Page 36 of 46 Formatted: Ref term, Font: Not Bold, English (U.S.)

Even though a single key pair may be used in multiple algorithms, the KMIP Specification has chosen to
 specify separate key formats for representing the asymmetric key pair for use in each algorithm. This
 approach keeps KMIP in line with the reference standards (e.g., NIST FIPS 186-3 [FIPS186-3], ANSI
 X9.42 [X9.42], etc) from which the key formats for DSA, DH, ECDSA, etc. are obtained and the best
 practice documents (e.g., NIST SP800-57 part 1 [SP800-57-1], NIST SP800-56A [SP800-56A], etc)
 which recommend that a key pair only be used for one purpose.

kmip-ug-1.0-cd-<mark>09</mark> Copyright © OASIS® <u>2010,2009. All Rights Reserved.</u>

5 November 2009

Formatted: Indent: Left: 0 pt, First line: 0 pt, Don't adjust space between Latin and Asian text

4 Deferred KMIP Functionality

The KMIP Specification is currently missing items that have been judged candidates for future inclusion in 1282 the specification. These items currently include: 1283 Registration of Clients. This would allow in-band registration and management of clients, which 1284 1285 currently may only be registered and/or managed using off-line mechanisms. Client-requested specification of additional clients that are allowed to use a key. This requires 1286 coordinated identities between the client and server, and as such, is deferred until registration of 1287 clients is addressed 1288 1289 Registration of Notifications. This would allow clients to specify, using an in-band mechanism, 1290 information and events that they wish to be notified of, and what mechanisms should be used for such notifications, possibly including the configuration of pushed cryptographic material. This 1291 1292 functionality would assume the Registration of Clients as a prerequisite. 1293 Key Migration. This would standardize the migration of keys from one HSM to another, using 1294 mechanisms already in the protocol or ones added for this purpose. Server to Server key management. This would extend the protocol to support communication 1295 between key management servers in different key management domains, for purposes of 1296 1297 exporting and importing cryptographic material and potentially policy information. 1298 Multiple derived keys. This would allow the creation of multiple derived keys from one or more 1299 input keys. Note, however, that the current version of KMIP provides the capability to derive 1300 multiple keys and initialization vectors by creating a Secret Data object and specifying a cryptographic length equal to the total length of the derived objects. 1301 1302 XML encoding. Expression of KMIP in XML rather than in tag/type/length/value may be 1303 considered for the future. 1304 Specification of Mask Generation Function. KMIP does not currently allow clients to specify the 1305 Mask Generation Function and assumes that encryption or signature schemes, such as OAEP or 1306 PSS, use MGF1 with the hash function as specified in the Cryptographic Parameters attribute. 1307 Client specification of MGFs may be considered for the future. 1308 Certificate creation without client-provided Certificate Request. This would allow clients to request 1309 the server to perform the Certify or Re-certify operation from the specified key pair IDs without 1310 providing a Certificate Request. 1311 Server monitoring of client status. This would enable the transfer of information about the client 1312 and its cryptographic module to the server. This information would enable the server to generate 1313 alarms and/or disallow requests from a client running component versions with known 1314 vulnerabilities 1315 Symmetric key pairs. Only a subset of the cryptographic usage bits of the Cryptographic Usage 1316 Mask attribute may be permitted for keys distributed to a particular client. KMIP does not currently 1317 address how to securely assign and determine the applicable cryptographic usage for a client. 1318 Hardware-protected attribute. This attribute would allow clients and servers to determine if a key may only be processed inside a secure cryptographic device, such as an HSM. If this attribute is 1319 1320 set, the key may only exist in cleartext within a secure hardware device, and all security-relevant attributes are bound to it in such a way that they may not be modified outside of such a secure 1321 1322 device. 1323 Alternative profiles for key establishment. Less capable end-clients may not be able to support 1324 TLS and should use a proxy to communicate with the key management system. The KMIP protocol does not currently support alternative profiles, nor does it allow end-clients relying on the 1325

proxy model to securely establish a key with the server.

1326

- Attribute mutation. The possibility for the server to use attribute values different than requested by the client if these values are not suitable for the server, and return these values in the response, instead of failing the request.
 Cryptographic Domain Parameters. KMIP allows a limited number of parameters to be specified during a Create Key Pair operation. Additional parameters may be considered for the future.
- Re-key support for other cryptographic objects. The Re-key operation is currently restricted to symmetric keys. Applying Re-key to other cryptographic objects, such as asymmetric keys and certificates, may be considered for the future.
- Certificate Suspension/Unsuspension. KMIP does not currently support certificate suspension (putting a certificate on hold) or unsuspension (removing a certificate from hold). Adding support for certificate suspension/unsuspension into KMIP may be considered for the future.
- Namespace registration. Establishing a registry for namespaces may be considered for the future.
- Registering extensions to KMIP enumerations. Establishing a registry for extensions to defined
 KMIP enumerations, such as in support of profiles specific to IEEE P1619.3 or other
 organizations, may be considered for the future.
- 1343 In addition to the functionality listed above, the KMIP TC is interested in establishing a C&A (certification 1344 and accreditation) process for independent validation of claims of KMIP conformance. Defining and
- 1345 establishing this process is a candidate for work by the KMIP TC after V1.0.

5 Implementation Conformance 1346

This document is intended to be informational only and as such has no conformance clauses. The conformance requirements for the KMIP <u>Specificationspecification</u> can be found in the "KMIP Specification" document itself, at the URL noted on the cover page of this document. 1347

1348 1349

kmip-ug-1.0-cd-<u>09</u> Copyright © OASIS® <u>2010.</u>2009. All Rights Reserved.

1350 A. Acronyms

1351	The follow	wing abbreviations and acronyms are used in this document:	
1352	3DES	- Triple Data Encryption Standard specified in ANSI X9.52	
1353	AES	- Advanced Encryption Standard specified in FIPS 197	
1354	ANSI	- American National Standards Institute	
1355	ARQC	- Authorization Request Cryptogram	
1356	ASCII	- American Standard Code for Information Interchange	
1357	CA	- Certification Authority	
1358	CBC	- Cipher Block Chaining specified in NIST SP 800-38A	
1359	CMC	- Certificate Management Messages over CMS specified in RFC 5275	
1360	CMP	- Certificate Management Protocol specified in RFC 4210	
1361	CRL	- Certificate Revocation List specified in RFC 5280	
1362	CRMF	- Certificate Request Message Format specified in RFC 4211	
1363	CVC	- Card Verification Code	
1364	DES	- Data Encryption Standard specified in FIPS 46-3	
1365	DEK	- Data Encryption Key	
1366	DH	- Diffie-Hellman specified in ANSI X9.42	
1367	FIPS	- Federal Information Processing Standard	
1368	GCM	- Galois/Counter Mode specified in NIST SP 800-38D	
1369	HMAC	- Keyed-Hash Message Authentication Code specified in FIPS 198-1	
1370	HSM	- Hardware Security Module	
1371	HTTP	- Hyper Text Transfer Protocol	
1372	HTTP(S)	- Hyper Text Transfer Protocol (Secure socket)	
1373	ID	- Identification	
1374	IP	- Internet Protocol	
1375	IPSec	- Internet Protocol Security	
1376	JKS	- Java Key Store	
1377	KEK	- Key Encryption Key	
1378	KMIP	- Key Management Interoperability Protocol	
1379	LTO4	- Linear Tape-Open 4	
1380	MAC	- Message Authentication Code	
1381	MD5	- Message Digest 5 Algorithm specified in RFC 1321	
1382	MGF	- Mask Generation Function	
1383	NIST	- National Institute of Standards and Technology	
1384	OAEP	- Optimal Asymmetric Encryption Padding specified in PKCS#1	
1385	PEM	- Privacy Enhanced Mail specified in RFC 1421	
	kmip-ug-1.0	0-04-09	18 N
1	Sobkinding	Choros <u>Evito</u> . An Agina Acourou.	г

8 March 201005 Page 41 of 46

1386	PGP	 Pretty Good Privacy specified in RFC 1991
1387	PKCS	- Public-Key Cryptography Standards
1388	POP	- Proof of Possession
1389	POSIX	- Portable Operating System Interface
1390	PSS	- Probabilistic Signature Scheme specified in PKCS#1
1391	RACF	- Remote Access Control Facility
1392	RSA	- Rivest, Shamir, Adelman (an algorithm)
1393	SHA	- Secure Hash Algorithm specified in FIPS 180-2
1394	SP	- Special Publication
		•
1395	SSL	- Secure Sockets Layer
1395 1396	S/MIME	- Secure Sockets Layer - Secure/Multipurpose Internet Mail Extensions
1395 1396 1397	S/MIME	- Secure Sockets Layer - Secure/Multipurpose Internet Mail Extensions - Transport Control Protocol
1395 1396 1397 1398	SSL S/MIME TCP TLS	Secure Sockets Layer Secure/Multipurpose Internet Mail Extensions Transport Control Protocol Transport Layer Security
1395 1396 1397 1398 1399	S/MIME S/MIME TCP TLS TTLV	Secure Sockets Layer Secure/Multipurpose Internet Mail Extensions Transport Control Protocol Transport Layer Security Tag, Type, Length, Value
1395 1396 1397 1398 1399 1400	S/MIME S/MIME TCP TLS TTLV URI	Secure Sockets Layer Secure/Multipurpose Internet Mail Extensions Transport Control Protocol Transport Layer Security Tag, Type, Length, Value Uniform Resource Identifier
1395 1396 1397 1398 1399 1400 1401	S/MIME TCP TLS TTLV URI X.509	 Secure Sockets Layer Secure/Multipurpose Internet Mail Extensions Transport Control Protocol Transport Layer Security Tag, Type, Length, Value Uniform Resource Identifier Public Key Certificate specified in RFC 5280

B. Acknowledgements

1016 Original Authors of the Initial contribution: 1017 Divid Babock, HP 1018 Staven Bade, IBM 1019 Patolo Bezoari, Net/opp 1110 Mathias Ejorkqvist, IBM 1121 Christan Cachin, IBM 1121 Jack Farwood, EMC 1121 Jack Harwood, EMC 1122 Valit Hubs, ISB 1122 Gen Jaquette, IBM 1122 Jack Harwood, EMC 1122 Jack Harwood, EMC 1122 Jack Harwood, EMC 1123 Gen Jaquette, IBM 1124 Jack Harwood, EMC 1125 Jack Harwood, EMC 1126 Hit Harwood, EMC 1128 Eliane Painer, IBM 1129 Jack Harwood, EMC 1129 Jack Harwood, EMC 1121 Jack Harwood, EMC 1122 Wark Schiller, HP 1123 Gen Jacuette, IBM 1124 Gen Jacuette, IBM	1404 1405	The following individuals have participated in the creation of this specification and are gratefully acknowledged:		
1407 David Babcock, HP 1408 Steven Bade, IBM 1409 Paolo Bezoari, NetApp 1410 Paolo Bezoari, NetApp 1411 Bruce Binnson, EMC 1412 Christein Cachin, EM 1413 Tony Crossman, Theles/TCpher 1414 Bruce Binnson, EMC 1415 Star Featler, HP 1416 Dia Grater, Theles/TCpher 1417 Jong Geater, Theles/TCpher 1418 Bob Griffin, EMC 1419 Robert Hass, IBM 1420 Timothy Hahn, IBM 1421 Jack Harwood, EMC 1422 Walt Hubbs, LSI 1423 Gen Jaquette, IBM 1424 Jeft Krautz, IBM 1425 Michael Michinsh, IBM 1426 Bran Metzger, HP 1427 Jack Harwood, EMC 1428 Bran Metzger, HP 1429 Bran Metzger, IBM 1420 Jon Tatan, EMC 1431 Subhash Sankuratripati, NetApp 1432 Jack Tarwood, NetApp 1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/NClipher 1435 John Tatan, EMC 1436 Participartis: 1437 Participartis:	1406	Original Authors of the initial contribution:		
1408 Steven Bade, IBM 1409 Paolo Bezani, NetApp 1411 Bruce Brinson, EMC 1412 Christen Cachin, IBM 1413 Tony Crossman. Thates/nCipher 1414 Stan Feather, HP 1415 Judy Funder, EMC 1416 Judy Funder, EMC 1417 Jon Ceater, Theses/nCipher 1418 Bob crifting, EMC 1419 Bob crifting, EMC 1420 Jack Harwood, EMC 1421 Jack Harwood, EMC 1422 Wait Hubis, LSI 1423 Glen Jaquette, IBM 1424 Jeff Kravitz, IBM 1425 Michael Micritosh, IBM 1426 Brain Mitzger, IPP 1427 Anthrony Nadalin, IBM 1428 Subhas Sankuratripati, NetApp 1429 Joe Pato, IP 1430 Rone Pawitzek, IBM 1431 Subhas Sankuratripati, NetApp 1432 Martin Skagen, Brocade 1433 Martin Skagen, Brocade 1444 Here Battek, Vendi, Inc. 1439 Steve Wierenga, HP 1444 Formatted: Check spelling and grammar 1445 Martin Skagen, Brocade 1446 Peter Battek, Vendi, Inc. 1447<	1407	David Babcock, HP		
1409 Paolo Bezcari, NetApp 1410 Mathias Björkyvst, IBM 1411 Bruce Brinson, EMC 1412 Christan Cachin, IBM 1413 Tony Crossman, Thales/nClpher 1414 Stan Feather, HP 1415 Indra Fitzgeraid, HP 1416 Judy Futiong, EMC 1417 Jon Ceater, Thales/nClpher 1418 Bob Griffin, EMC 1419 Robert Haas, IBM 1420 Tinothy Hain, IBM 1421 Jack Harwood, EMC 1422 Wait Hubs, LSI 1423 Glen Jaquette, IM 1424 Bian Metzger, HP 1425 Hair Mathias, Biorky, IBM 1426 Hair Mateger, HP 1430 René Parlmer, IBM 1429 Joe Pato, HP 1430 René Parlmer, IBM 1429 Joe Pato, HP 1430 Karla Thomas, Brocade 1444 Marcus Streets, Thales/nClpher 1431 Streew Wirenga, HP 1444 Elaine Parlmer, IBM 1438 Streew Wirenga, HP 1444 Marcus Streets, Thales/nClpher 1454 Hairn Skagen, Streets, Thales/nClpher 1454 Here Batrick, Verafin, Inc. 1454	1408	Steven Bade, IBM		
1410 Mathias Björkqvist, libM 1411 Bruce Brinson, EMC 1412 Christian Cachin, IBM 1413 Tony Crossman, Thales/nClpher 1414 Stan Feather, HP 1415 Indra Fitzgerald, HP 1416 Judy Furiong, EMC 1417 Jon Geater, Theles/nClpher 1418 Bob Griffin, EMC 1419 Robert Haas, IBM 1421 Track Hanvoot, EMC 1422 Wait Hubis, ISJ 1423 Gen, Jaquette, IBM 1424 Jeff Kravitz, IBM 1425 Michael McIntosh, IBM 1426 Baine Palmer, IBM 1427 Anthony Nadalin, IBM 1428 Ealine Palmer, IBM 1429 Jeef Kravitz, IBM 1431 Subhash Sankuratripati, NetApp 1432 Martin Skagen, Brocade 1433 Martin Skagen, Brocade 1444 Gordon Arnold, IBM 1439 Rever Boack, Thales nClpher 1444 Elaine Barker, NIST 1445 Peter Barko, Venafi, Inc. 1446 Mathew Ball, CipCe Corporation 1447 Kelley Burgin, National Security Agency 1448 Kalley Cooparation 1444 Elaine Barker, NIST	1409	Paolo Bezoari, NetApp		
1411 Bruce Brinson, EMC 1412 Christian Cachin, IBM 1413 Tony Crossman, Thales/nClpher 1414 Stan Feather, HP 1415 Indra Fitzgeraid, HP 1416 Judy Furiong, EMC 1417 Jon Geater, Thales/nClpher 1418 Bob Griffin, FMC 1419 Robert Haas, IBM 1420 Timothy Hain, IBM 1421 Jack Harwood, EMC 1422 Wait Hubs, ISI 1423 Glen Jaquette, IBM 1424 Jef Kravitz, IBM 1425 Mitchael McInitosh and 1426 Hichael McInitosh and 1427 Anthro My Medialin, IBM 1428 Jon Pato, IPP 1430 Rene Palmer, IBM 1431 Subhash Sankreathgait, NetApp 1433 Marko Stalen, EnCd 1434 Marcus Streets, ThalesinCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Todd Annold, IBM 1438 Steve Wierenga, HP 1444 Formatted: Check spelling and gammar	1410	Mathias Björkgvist, IBM		
1412 Christian Cachin, IBM 1413 Tony Crossman, Thales/Clipher 1414 Stan Feather, HP 1415 Indra Fitzgeraid, HP 1416 Judy Furlong, EMC 1417 Jon Geater, Thales/Clipher 1418 Bob Griffin, EMC 1419 Robert Haas, IBM 1420 Timothy Hahn, IBM 1421 Jack Harwood, EMC 1422 Walt Hubis, LSI 1423 Gein Jaquette, IBM 1424 Jeff Kravitz, IBM 1425 Michael Michinosh, IBM 1426 Brian Metzger, HP 1427 Hartonos, IBM 1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1430 Rene Pawitzek, IBM 1431 Subhash Sankuratingal, NetApp 1432 Mark Stoller, HP 1433 Karia Thomas, Brocade 1434 Marcus Streets, ThalesmCipher 1435 John Tatian, EMC 1436 Karia Thomas, Brocade 1437 Maths Outcoit, IBM 1438 Rober, Nill 1441 Pato	1411	Bruce Brinson, EMC		
1413 Tory Crossman, Thales/nCipher 1414 Stan Feather, HP 1415 Indra Fitzgerald, HP 1416 Judy Furinog, EMC 1417 Jon Geater, Thales/nCipher 1418 Bob Griffin, EMC 1419 Robert Haas, IBM 1420 Timothy Hahn, IBM 1421 Jack Harwood, EMC 1422 Walt Hubis, LSI 1423 Glen Jaquette, IBM 1424 Jeff Kravitz, IBM 1425 Michael Molnosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadalin, IBM 1428 Elline Palmer, IBM 1429 Joe Pato, HP 1430 René Pawiltzek, IBM 1431 Subhash Sankuratripati, NetApp 1433 Marcus Streets, Thales/nCipher 1434 Marcus Streets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Sullier, HP 1438 Steve Wierenga, HP 1439 Participants: 1441 Mathw Ball, Oracle Corporation	1412	Christian Cachin, IBM		
1414 Staf Feather, HP 1415 Indra Fitzgradi, HP 1416 Judy Furiong, ENC 1417 Jon Geater, Theles/nCipher 1418 Bob Griffin, EMC 1419 Robert Haas, IBM 1420 Timothy Hahn, IBM 1421 Jack Harwood, EMC 1422 Walt Hubis, LSI 1423 Glen Jaquette, IBM 1424 Jeft Kraviz, IBM 1425 Mitchael Moltosh, IBM 1426 Brian Metzger, HP 1427 Anthory Nadain, IBM 1428 Elane Palmer, IBM 1429 Joe Pato, HP 1430 Renef Pawlitzek, IBM 1431 Subhash Sankurathrpati, NetApp 1432 Mark Schiller, HP 1433 Martin Skage, Brocade 1434 Marcus Streets, Thalesin/Cipher 1435 Narko Vukoli, IBM 1438 Steve Wierenga, HP 1439 Marko Vukoli, IBM 1439 Marko Vukoli, IBM 1439 Marko Vukoli, IBM 1439 Marko Vukoli, IBM 1440 Deordon A	1413	Tony Crossman. Thales/nCipher		
1415 India Filzgeräid, HP 1416 Judy Furlong, EMC 1417 Jon Geater, Thales/nClpher 1418 Bob Griffin, EMC 1419 Robert Haas, IBM 1420 Timothy Hahn, IBM 1421 Jack Hawood, EMC 1422 Walt Hubis, LSI 1423 Gien Jaquette, IBM 1424 Jeff Kravitz, IBM 1425 Michael Molnosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadain, IBM 1428 Ellane Palmer, IBM 1429 Joe Pato, HP 1421 Subhash Sankuratripati, NetApp 1433 Marcus Streets, Thales/nClpher 1434 Marcus Streets, Thales/nClpher 1435 Sibew Witerang, HP 1436 Bitew Witerang, HP 1437 Markus Sholper, HP 1438 Stew Witerang, HP 1439 Participants: 1441 Mathwe Ball, Oracle Corporation 1442 Orad Arnoid, IBM 1444 Reine Barkty, Netani, Inc. 1445 Peter Barus, Kreins I, Inc.	1414	Stan Feather, HP		
1416 Judy Funding, EMC 1417 Jon Geater, Thales/nClpher 1418 Bob Griffin, EMC 1419 Robert Haas, IBM 1420 Timothy Hahn, IBM 1421 Jack Harvood, EMC 1422 Walt Hubis, LSI 1423 Glen Jacquette, IBM 1424 Jeff Kravitz, IBM 1425 Michael McInoba, IBM 1426 Brian Metzger, HP 1427 Anthory Natalin, IBM 1428 Beine Paimer, IBM 1429 Joe Pato, HP 1431 Subhash Sankurajneli, NetApp 1432 Join Tatan, EMC 1433 Matricus Streest, Thales/nClpher 1444 John Tatan, EMC 1435 John Tatan, EMC 1440 Matricus Biocrade 1441 Gardon Arnold, IBM 1442 TodA Arnold, IBM 1443 Matricus Streew Wierenga, HP 1444 Haine Barker, NIST 1445 Peter Bartok, Venati, Inc. 1446 Mathas Boinger, National Security Agency 1447 Graydon Dotson, Lexmark Intemational Inc.	1415	Indra Fitzgerald. HP		
1417 Jon Ceater, Thales/NClpher 1418 Bob Griffin, EMC. 1419 Robert Haas, IBM 1420 Timothy Hahn, IBM 1421 Jack Harvood, EMC. 1422 With Hubb, ISI 1423 Gen Jaquette, IBM 1424 Jeft Kravtz, IBM 1425 Michael Mchinosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadain, IBM 1428 Elaine Palmer, IBM 1429 Joo Pato, HP 1430 René Pawitzek, IBM 1431 Subhash Sankuratripati, NetApp 1433 Martin Skagen, Brocade 1434 Marto Studei, IBM 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Yukoli, IBM 1438 Steve Wierenga, HP 1449 Jon Cattan, IBM 1440 Gordon Arnold, IBM 1441 Gordon Arnold, IBM 1442 Participants: 1444 Mathues Bain, Natonal Security Agency 1445 Peter Barke, Yoardi, In.c. 1446	1416	Judy Furlong, EMC		
1418 Bob Griffin, EMC 1419 Robert Haas, IBM 1420 Timothy Hahn, IBM 1421 Jack Harvood, EMC 1422 Walt Hubis, LSi 1423 Glen Jaquette, IBM 1424 Jeft Kravitz, IBM 1425 Michael McInotsh, IBM 1426 Brian Metzger, HP 1427 Anthory Nadalin, IBM 1428 Elaine Palmer, IBM 1429 Joe Palo, IHP 1430 Rende Pawitizek, IBM 1441 Subhash Sankutiripati, NetApp 1442 Mark Schlifer, HP 1431 Subhash Sankutiripati, NetApp 1443 Macrus Streest, Thalesn(C)pher 1444 Marcus Streest, Thalesn(C)pher 1455 Join Tatan, EMC 1464 Marcus Streest, Thalesn(C)pher 1473 Natros Streest, Thalesn(C)pher 1474 Marcus Streest, Thalesn(C)pher 1475 Marcus Streest, Thalesn(C)pher 1476 Marcus Streest, Thalesn(C)pher 1477 Marcus Streest, Thalesn(C)pher 1478 Natros Streest, Thalesn(C)pher	1417	Jon Geater. Thales/nCipher		
1419 Robert Haas, IBM 1420 Timothy Han, IBM 1421 Jack Harwood, EMC 1422 Walt Hubis, IS1 1423 Glen, Jaquette, IBM 1424 Jeff Kravitz, IBM 1425 Michael McIntosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadalin, IBM 1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1430 Rene Pawitzek, IBM 1431 Subhash Sankurathpali, NetApp 1432 Martin Skagen, Brocade 1433 Martins Kagen, Brocade 1434 Martous Streets, Thales/nClipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukilo, IBM 1438 Steve Wierenga, HP 1441 Gordon Antold, IBM 1435 John Tattan, Idone, Corporation 1444 Hathew Ball, Oracle CorporationSun-Microsystems 1444 Feirmatted: Check spelling and grammar 1445 Pater Bardx, Vanafi, Inc. 1446 Relieg Stating, National Security Agency 144	1418	Bob Griffin, EMC		
1420 Timothy Hain, IEM 1421 Jack Harwood EMC 1422 Walt Hubis, LSI 1423 Glen Jaquette, IBM 1424 Jeff Kravitz, IBM 1425 Michael McIntosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadalin, IBM 1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1420 Mark Schlifter, HP 1431 Subhash Sankuratipati, NetApp 1432 Mark Schlifter, HP 1433 Marin Skagen, Brocade 1434 Marcus Streest, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vucidi, IBM 1438 Karla Thomas, Brocade 1441 Godron Anold, IBM 1442 Todd Arnold, IBM 1444 Mathew Ball, Gracle Corporation Sun Microsystems 1444 Kelley Burgin, National Security Agency 1445 Peter Barko, Venafi, Inc. 1446 Kelley Burgin, National Security Agency 1447 Kelley Burgin, National Security Agency	1419	Robert Haas, IBM		
1421 Jack Harwood, ENC 1422 Walt Hubis, LSI 1423 Glen Jaquette, IBM 1424 Jeff Kraviz, IBM 1425 Michael McInosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadalin, IBM 1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1430 René Pawitzek, IBM 1431 Subhash Sankuratripati, NetApp 1433 Marcus Streets, ThalesinCipher 1434 John Tattan, EMC 1435 Marko Vukolić, IBM 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1440 Mite Allen, PCP Corporation 1441 Todd Arnold, IBM 1442 Todd Arnold, IBM 1443 Mathwe Ball, Oracle Corporation Sun Microsystems 1444 Kelley Burgin, National Security Agency 1445 Peter Bartok, Venafi, Inc. 1446 Mathwe Ball, Oracle Corporation 1447 Kelley Burgin, National Security Agency 1448 Kelley Burgin, National Security Agency	1420	Timothy Hahn, IBM		
1422 Wait Hubis LSI 1423 Glen Jaquette, IBM 1424 Jef Kravitz, IBM 1425 Michael McInissh, IBM 1426 Brian Mctzger, HP 1427 Anthony Nadalin, IBM 1428 Elaine Paimer, IBM 1429 Joe Pato, HP 1420 René Pawitzek, IBM 1421 Joe Pato, HP 1422 Joe René Pawitzek, IBM 1423 Joe Pato, HP 1430 René Pawitzek, IBM 1431 Subhash Sankuratripati, NetApp 1432 John Tattan, EMC 1433 Karla Thomas, Brocade 1434 Marcus Streets, ThalesinCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1441 Todd Arnoid, IBM 1442 Todd Arnoid, IBM 1444 Todd Arnoid, IBM 1445 Peter Bartok, Venafi, Inc. 1446 Methew Ball, Oracle Corporation 1447 Kelley Burgin, National Security Agency <t< td=""><td>1421</td><td>Jack Harwood, EMC</td><td></td><td></td></t<>	1421	Jack Harwood, EMC		
1423 Glen Jaquette, IBM 1424 Jeff Kraviz, IBM 1425 Michael McIntosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadalin, IBM 1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1430 René Pawlitzek, IBM 1431 Subhash Sankuratripati, NetApp 1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Steve Wierenga, HP 1440 Mike Allen, PCP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Patricipants: 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Matthies Bliotravist Iperker, Indexin, Iternational Inc. 1447 Kelley Burgin, National Security Agency 1448 Relley, Bray, SafeNet, Inc. 1449 Jon Callas, Hewletth PackardHP, <td< td=""><td>1422</td><td>Walt Hubis, LSI</td><td></td><td></td></td<>	1422	Walt Hubis, LSI		
1424 Jeff Kravitz, IBM 1425 Michael McIntosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadalin, IBM 1428 Elaine Patmer, IBM 1429 Joe Pato, HP 1420 René Pawitzek, IBM 1421 Subbash Sankuratipati, NetApp 1422 Markin Skagen, Brocade 1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/nCipher 1435 John Tatina, EMC 1436 Karla Thomas, Brocade 1437 Markolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todi Arnold, IBM 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BjorkwyisBjorkwyist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kalene Barker, NIST 1449 Check spelling and grammar 1444 Formatted: Check spelling and grammar 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BjorkwyisBjorkwyist, IBM 1447 Keivin Bocek, Thales e-Security 1448 Ka	1423	Glen Jaguette, IBM		
1425 Michael Mcintosh, IBM 1426 Brian Metzger, HP 1427 Anthony Nadalin, IBM 1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1430 René Pawlitzek, IBM 1431 Subbash Sankuratripati, NetApp 1432 Mark Schiller, HP 1433 Martin Skagen, Brocade 1434 Marcus Streets, ThalesinCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Toid Arnold, IBM 1443 Matthew Bail, Oracle CorporationSun-Microsystems 1444 Elaine Barker, NIST 1445 Peter Barkok, Venafi, Inc. 1446 Mathias Bjorkdysiti, IBM 1447 Keiley Burgin, National Security Agency 1448 Keiley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1451 Graydon Dodson, Lexmark International Inc. <td>1424</td> <td>Jeff Kravitz, IBM</td> <td></td> <td></td>	1424	Jeff Kravitz, IBM		
1426 Brian Metzger, H ^P 1427 Anthony Nadalin, IBM 1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1430 René Pawlitzek, IBM 1431 Subhash Sankuratripati, NetApp 1432 Mark Schiller, HP 1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Mark Schiller, HP 1438 Steve Wierenga, HP 1439 Participants: 1440 Gordon Amold, IBM 1441 Gordon Amold, IBM 1442 Todd Amold, IBM 1443 Hatnew Ball, Oracle CorporationSun-Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias Biotkery Surgin, National Security Agency 1447 Kelley Burgin, National Security Agency 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn,	1425	Michael McIntosh. IBM		
1427 Anthony Nadalin, IBM 1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1430 René Pawiltzek, IBM 1431 Subhash Sankuratripati, NetApp 1432 Mark Schiller, HP 1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun-Microeystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathew Ball, Oracle Corporation Security Agency 1447 Keiley Burgin, National Security Agency 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation Inc. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, Safelvet, Inc. 1454 Stan Feather, Hewlett-PackardHP, <tr< td=""><td>1426</td><td>Brian Metzger, HP</td><td></td><td></td></tr<>	1426	Brian Metzger, HP		
1428 Elaine Palmer, IBM 1429 Joe Pato, HP 1430 René Pawitzek, IBM 1431 Subhash Sankuratripati, NetApp 1432 Mark Schiller, HP 1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Schiller, HP 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1444 Elaine Barker, NIST 1445 Peter Barker, NIST 1446 Mathias Björkqvist, IBM 1447 Kevin Bock, Thales -Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Graydon Dodson, Lexmark International Inc. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP,	1427	Anthony Nadalin, IBM		
1429 Joe Pato, HP 1430 René Pawlitzek, IBM 1431 Subhash Sankuratripati, NetApp 1432 Mark Schiller, HP 1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukoló, IBM 1438 Steve Wierenga, HP 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Matthew Bail, Oracle CorporationSun Microsystems 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1449 Jon Callas, PGP Corporation 1450 Tord Ufford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-Packard+P. 1454	1428	Elaine Palmer, IBM		
1430 René Pawlitzek, IBM 1431 Subhash Sankuratripati, NetApp 1432 Mark Schiller, HP 1433 Martin Skagen, Brocade 1434 Marcus Strets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Schiller, HP 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Barkot, Venafi, Inc. 1446 Mathias BiorkayistBjorkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP, 16	1429	Joe Pato, HP		
1431 Subhash Sankuratripati, NetApp 1432 Mark Schiller, HP 1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PCP Corporation 1441 Gordon Armold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun-Microsystems 1444 Elaine Barker, NIST 1445 Peter Barker, NIST 1446 Mathias Björkqvist IBM 1447 Kevin Bock, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Pather, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Inc. 1455 Formatted: Check spelling and grammar Formatted: Check spelling and	1430	René Pawlitzek, IBM		
1432 Mark Schiller, HP 1433 Martin Skagen, Brocade 1434 Marcus Streets, Theles/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Gordon Amold, IBM 1441 Todd Arnold, IBM 1442 Todd Arnold, IBM 1444 Elaine Barker, NIST Peter Bartok, Venafi, Inc. Formatted: Check spelling and grammar 1444 Keiley Burgin, National Security Agency 1445 Graydon Doson, Lexmark International Inc. 1445 Chris Dun, SafeNet, Inc. 1450 Tom Clifford, Symantec Corp. 1451 Graydon Doson, Lexmark International Inc. 1452 Chris Dun, SafeNet, Inc. 1453 Paul Earsy. SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP. 1455 Rinip-ug-1.0-cd-02 1456 Kenip-ug-1.0-cd-02 1457 Copyright © Ox8150 <u>Ported</u> -All Rights Reserved. 1458 Stan Feather, Hewlett-PackardHP. <tr< td=""><td>1431</td><td>Subhash Sankuratripati, NetApp</td><td></td><td></td></tr<>	1431	Subhash Sankuratripati, NetApp		
1433 Martin Skagen, Brocade 1434 Marcus Streets, Thales/nC/pher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun-Microsystems 1444 Elaine Barker, NIST 1444 Peter Bartok, Venafi, Inc. 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BjorkqvistBiorkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Paul Earsy, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP, 1455 Indra Fitzgeraid, Hewlett-PackardHP, 1454 Stan Feather, Hewlett-PackardHP, 1455 Indra Fitzgeraid, Hewlett-PackardHP	1432	Mark Schiller, HP		
1434 Marcus Streets, Thales/nCipher 1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BlorkovistBjorkevist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP 1455 Formatted: Check spelling and grammar 1454 Stan Feather, Hewlett-PackardHP 1455 Formatted: Check spelling and grammar <td>1433</td> <td>Martin Skagen, Brocade</td> <td></td> <td></td>	1433	Martin Skagen, Brocade		
1435 John Tattan, EMC 1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BjörkqvistBjörkqvistBjörkqvistBiBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Indra Fitzgeraid, Hewlett-PackardHP 1455 Indra Fitzgeraid, Hewlett-PackardHP 1454 Kelley Oolgoeon- All Rights Reserved. 1455 Stan Feather, Hewlett-PackardHP 1456 Kenie Oolsise 2010;2009- All Rights Reserved.	1434	Marcus Streets. Thales/nCipher		
1436 Karla Thomas, Brocade 1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Gordon Arnold, IBM 1441 Todd Arnold, IBM 1442 Todd Arnold, IBM 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BjörkqvistBjörkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP, 1455 Indra Fitzgeraid, Hewlett-PackardHP, 1454 Kaley 2010_2009. All Rights Reserved.	1435	John Tattan, EMC		
1437 Marko Vukolić, IBM 1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias Bijorkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP, 1455 Indra Fitzgerald, Hewlett-PackardHP, 1456 Indra Fitzgerald, Hewlett-PackardHP, 1457 Page 43 of 46	1436	Karla Thomas, Brocade		
1438 Steve Wierenga, HP 1439 Participants: 1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BjörkqvistBjörkqvistBjörkqvist IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1455 Stan Feather, Hewlett-PackardHP 1456 Envirot.0-09 1457 Stan Feather, Hewlett-PackardHP 1458 Copyright © OASIS® 2010_2009. All Rights Reserved.	1437	Marko Vukolić IBM		
1439 Participants: 1440 Gordon Arnold, IBM 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BjörkqvistBjørkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1455 Stan Seaverd. Page 43 of 46	1438	Steve Wierenga, HP		
1440 Mike Allen, PGP Corporation 1441 Gordon Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias Björkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP, 1455 Indra Fitzgerald, Hewlett-PackardHP, 1455 Indra Fitzgerald, Hewlett-PackardHP, 1455 Stan Feather, Hewlett-PackardHP, 1455 Indra Fitzgerald, Hewlett-PackardHP, 1455 November 2009 Copyright © OASIS® 2010,2009- All Rights Reserved. Page 43 of 46	1439	Participants:		
1441 Cordon Arnold, IBM 1442 Todd Arnold, IBM 1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias BjörkqvistBjorkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Pather: Hewlett-PackardHP 1454 Stan Feather, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1455 OASIS® 2010, 2009. All Rights Reserved.	1440	Mike Allen, PGP Corporation		
1442 Todd Arnold, IBM 1443 Matthew Ball, Oracle CorporationSun Microsystems 1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias Björkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP, 1455 Indra Fitzgerald, Hewlett-PackardHP, 1456 Indra Fitzgerald, Hewlett-PackardHP, 1457 Page 43 of 46	1441	Gordon Arnold IBM	•	Formatted: Check spelling and grammar
Hatting Matthew Ball, Oracle CorporationSun Microsystems Formatted: Body Text, Indent: First line: 36 pt 1444 Elaine Barker, NIST Formatted: Body Text, Indent: First line: 36 pt 1444 Elaine Barker, NIST Formatted: Body Text, Indent: First line: 36 pt 1444 Elaine Barker, NIST Formatted: Body Text, Indent: First line: 36 pt 1445 Peter Bartok, Venafi, Inc. Formatted: Check spelling and grammar 1446 Mathias BjörkqvistBjorkqvist, IBM Formatted: Check spelling and grammar 1447 Kevin Bocek, Thales e-Security Formatted: Check spelling and grammar 1448 Kelley Burgin, National Security Agency Formatted: Check spelling and grammar 1450 Tom Clifford, Symantec Corp. Formatted: Check spelling and grammar 1451 Graydon Dodson, Lexmark International Inc. Formatted: Check spelling and grammar 1452 Chris Dunn, SafeNet, Inc. Formatted: Check spelling and grammar 1455 Indra Fitzgerald, Hewlett-PackardHP Formatted: Check spelling and grammar 1455 Indra Fitzgerald, Hewlett-PackardHP Formatted: Check spelling and grammar 1455 Copyright © OASIS® 2010, 2009. All Rights Reserved. Page 43 of 46	1442	Todd Arnold IBM		Tormatted. Cricck spenning and grammar
1444 Elaine Barker, NIST 1445 Peter Bartok, Venafi, Inc. 1446 Mathias Björkqvist Bjorkqvist, IBM 1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1456 Indra Fitzgerald, Hewlett-PackardHP 1457 Kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009: All Rights Reserved. Page 43 of 46	1443	Matthew Ball. Oracle CorporationSun Microsystems		Formatted: Body Text, Indent: First line: 36
1445 Peter Bartok, Venafi, Inc. Formatted: Check spelling and grammar 1446 Mathias Björkqvist, IBM	1444	Elaine Barker, NIST		pi
Hathias BjörkqvistBjörkqvistBjörkqvistBiorkqvistBjörkqvistBjjörkqvistBjörkq	1445	Peter Bartok, Venafi Inc		Formatted: Check spelling and grammar
1447 Kevin Bocek, Thales e-Security 1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1456 Copyright © OASIS® 2010, 2009. All Rights Reserved.	1446	Mathias BiörkovistBiorkovist IBM		Formatted: Check spelling and grammar
1448 Kelley Burgin, National Security Agency 1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1456 Tom Clifford, Summar Formatted: Check spelling and grammar Formatted: Check spelling and grammar Kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43 of 46	1447	Kevin Bocek, Thales e-Security		Tormatted: check spennig and grammar
1449 Jon Callas, PGP Corporation 1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1456 Stan Feather, Hewlett-PackardHP 1457 Formatted: Check spelling and grammar Kmip-ug-1.0-cd-09 18 March 201005 Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43 of 46	1448	Kelley Burgin National Security Agency		
1450 Tom Clifford, Symantec Corp. 1451 Graydon Dodson, Lexmark International Inc. 1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP. 1455 Indra Fitzgerald, Hewlett-PackardHP. 1456 Stan Feather, Hewlett-PackardHP. 1457 Indra Fitzgerald, Hewlett-PackardHP. 1458 Stan Feather, Hewlett-PackardHP. 1459 Indra Fitzgerald, Hewlett-PackardHP. 1450 Yes and Y	1449	Ion Callas PGP Corporation		
Instruction	1450	Tom Clifford Symantec Corp		
1452 Chris Dunn, SafeNet, Inc. 1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP 1455 Indra Fitzgerald, Hewlett-PackardHP 1456 Kmip-ug-1.0-cd-09 Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43 of 46	1451	Gravdon Dodson L exmark International Inc		
1453 Paul Earsy, SafeNet, Inc. 1454 Stan Feather, Hewlett-PackardHP, 1455 Indra Fitzgerald, Hewlett-PackardHP, 1456 Formatted: Check spelling and grammar 1457 Formatted: Check spelling and grammar 1458 Indra Fitzgerald, Hewlett-PackardHP, 1459 Indra Fitzgerald, Hewlett-PackardHP, 1450 Page 43 of 46	1452	Chris Dunn SafeNet Inc		
1454 Stan Feather, Hewlett-PackardHP, Formatted: Check spelling and grammar 1455 Indra Fitzgerald, Hewlett-PackardHP, Formatted: Check spelling and grammar kmip-ug-1.0-cd-09 18 March 201005 5 November 2009 Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43 of 46 Page 43 of 46	1453	Paul Earsy. SafeNet. Inc.		
1455 Indra Fitzgerald, Hewlett-Packard HP Formatted. Creck spelling and grammar kmip-ug-1.0-cd-09 18 March 201005 5 November 2009 Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43 of 46	1454	Stan Feather, Hewlett-PackardHP		Formatted: Check spelling and grammar
kmip-ug-1.0-cd-09 18 March 201005 5 November 2009 Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43 of 46	1455	Indra Fitzgerald, Hewlett-Packard		
kmip-ug-1.0-cd-09 18 March 201005 5 November 2009 Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43 of 46				Formatted: Check spelling and grammar
Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43 of 46		kmip-ug-1.0-cd- <u>09 18 March 20</u>	01005	5 November 2009
		Copyright © OASIS® 2010,2009. All Rights Reserved. Page 43	of 46	

4450	Alon Friedell, OpfeNet Jac		
1450	Alan Frindell, Salenet, Inc.		
1457	Judith Fullong, EMC Corporation		
1458	Jonathan Geater, Thales e-Security		
1459	Robert Griffin, EMC Corporation		
1460	Robert Haas, IBM		
1461	Inomas Hardjono, M.I.I.		
1462	Kurt Heberlein, 3PAR, Inc.	6	
1463	Marc Hocking, BeCrypt Ltd.	- 1	Formatted: Check spelling and grammar
1464	Larry Hoter, Emulex Corporation	` {	Formatted: Body Text, Indent: First line: 36
1465	Brandon Hoff, Emulex Corporation		pt
1466	Walt Hubis, LSI Corporation		
1467	Wyllys Ingersoll, Oracle CorporationSun Microsystems	- 1	Formatted: Check spelling and grammar
1468	Jay Jacobs, Target Corporation		
1469	Glen Jaquette, IBM		
1470	Scott Kipp, Brocade Communications Systems, Inc.		
1471	David Lawson, Emulex Corporation		
1472	Hal Lockhart, Oracle Corporation		
1473	Robert Lockhart, Thales e-Security	{	Formatted: Check spelling and grammar
1474	Shyam Mankala, EMC Corporation	- 1	Formatted: Body Text Indent: First line: 36
1475	Upendra Mardikar, PayPal Inc.		pt
1476	Marc Massar, Individual		Farmantina di Charak analijina and ananyara
1477	Don McAlister, Associate		Formatted: Check spelling and grammar
1478	Hyrum Mills, Mitre Corporation		Formatted: Body Text, Indent: First line: 36
1479	Bob Nixon, Emulex Corporation	ļ	pt
1480	Landon <u>Curt</u> Noll, Cisco Systems, Inc.	{	Formatted: Check spelling and grammar
1481	René Pawlitzek, IBM	- 1	Formatted: Body Text Indent: First line: 36
1482	Rob Philpott, EMC Corporation		pt
1483	Scott Rea, Individual	$\langle \rangle$	Forments d. Check and line and memory
1484	Bruce Rich, IBM	$\langle \rangle$	Formatted: Check spelling and grammar
1485	Scott Rotondo, Oracle CorporationSun Microsystems	V. I	Formatted: German (Switzerland), Check
1486	Saikat Saha, Vormetric, Inc.	λļ	spelling and grammar
1487	Anil Saldhana, Red Hat	N (Formatted: Check spelling and grammar
1488	Subhash Sankuratripati, NetApp	```	Formatted: Check spelling and grammar
1489	Mark Schiller, Hewlett-PackardHP	. \}	
1490	Jitendra Singh, Brocade Communications Systems, Inc.	X	Formatted: Body Text, Indent: First line: 36
1491	Servesh Singh, EMC Corporation	, it	pi
1492	Terence Spies, Voltage Security	1	Formatted: Check spelling and grammar
1493	Sandy Stewart, Oracle CorporationSun Microsystems	if	Formatted: Check spelling and grammar
1494	Marcus Streets, Thales e-Security	1.17	Formatted: Pody Text Indent: First line: 26
1495	Brett Thompson, SafeNet, Inc.	λ	nt
1496	Benjamin Tomhave, Individual	1	
1497	Sean Turner, IECA, Inc.	γv	Formatted: Check spelling and grammar
1498	Paul Turner, Venafi, Inc.	1.1	Formatted: Check spelling and grammar
1499	Marko <u>Vukolić</u> Vukolić, IBM	Ϋ́	Formatted: Body Text, Indent: First line: 36
1500	Rod Wideman, Quantum Corporation	- 1	pt
1501	Steven Wierenga, Hewlett-PackardHP	Ì	Formatted: Check spelling and grammar
1502	Peter Yee, EMC Corporation	$\left \right\rangle$	
1503	Krishna Yellepeddy, IBM	\cdot	Formatted: Check spelling and grammar
1504	Peter Zelechoski, Election Systems & SoftwareAssociate	Í	Formatted: Check spelling and grammar
1505	Grace Zhang, Skyworth TTG Holdings Limited		Formatted: Check spelling and grammar

C. Revision History 1506

Revision	Date	Editor	Changes Made	
ed-0.98	2009-04-29	Indra Fitzgerald	Initial conversion of input document to OASIS format.	
ed-0.98	2009-07-28	Indra Fitzgerald	Added clarifications, examples, and deferred items.	
ed-0.98	2009-09-08	Indra Fitzgerald	Added approved proposals and incorporated Elaine Barker's comments.	
ed-0.98	2009-09-23	Indra Fitzgerald	Removed KMIP Profiles section and incorporated the Interoperable Key Naming for Tape proposal.	
ed-0.98	2009-09-24	Indra Fitzgerald	Removed the Conformance section; added additional Certificate Request and POP text to Certify and Re-certify; added the Revocation Reason Codes section.	
draft-01	2009-10-07	Indra Fitzgerald	Incorporated the Certificate Renewal, Update, Re-key proposal, the Key Encoding proposal; removed normative words "must", "shall", "required", "will", and "can"; added Create Key Pair example; updated the references and acronyms list; incorporated comments from RobertH and SubhashS; updated the Authentication section; added minor edits and clarifications.	
draft-02	2009-10-09	Indra Fitzgerald	Incorporated Rod Wideman's comments on the language. Changed the heading indentation, paragraph style, and list styles according to the OASIS template guidelines. Added additional references. Replaced the TBDs. Added a use- case for registering a wrapped key as an opaque cryptographic object.	
draft-03	2009-10-21	Indra Fitzgerald	Added the list of participants to Appendix B. Clarified the Authentication section (section 3.1) and added examples. Modified the title page. Performed minor editorial changes.	
draft-04	2009-11-06	Indra Fitzgerald	Incorporated Elaine's comments.	
			This is the tentative revision for public review.	
draft-05	2009-11-09	Indra Fitzgerald	Minor edits to the reference sections.	
<u>draft-06</u>	<u>2010-02-24</u>	Indra Fitzgerald	Addressed public review comments. Clarified how templates work (section 3.6). Added Judy Furlong's proposal on using the same asymmetric key pair in multiple algorithms (section 3.34).	
draft-07	<u>2010-03-04</u>	Indra Fitzgerald	Clarified that the Destroy operation cannot destroy Active objects (section 3.5).	
draft-08	<u>2010-03-17</u>	Indra Fitzgerald	Added the Server Policy section (2.5). Added the Credential section (3.1.1) to the Authentication section. Replaced SSL/TLS with TLS. Updated the participant list. Other minor edits.	
draft-09	<u>2010-03-18</u>	Indra Fitzgerald	Renamed Role Type to Key Role Type. Updated the participant list.	

kmip-ug-1.0-cd-<u>09</u> Copyright © OASIS® <u>2010,</u>2009. All Rights Reserved.

<u>18 March 2010</u> Page 45 of 46

1507

Page 4: [1] Change	Unknown	
Field Code Changed		
Page 4: [1] Change	Unknown	
Field Code Changed		
Page 4: [2] Change	Unknown	
Field Code Changed		
Page 4: [2] Change	Unknown	
Field Code Changed		
Page 4: [3] Change	Unknown	
Field Code Changed		
Page 4: [3] Change	Unknown	
Field Code Changed		
Page 4: [4] Change	Unknown	
Field Code Changed		
Page 4: [4] Change	Unknown	
Field Code Changed		
Page 4: [5] Change	Unknown	
Field Code Changed		
Page 4: [5] Change	Unknown	
Field Code Changed		
Page 4: [6] Change	Unknown	
Field Code Changed		
Page 4: [6] Change	Unknown	
Field Code Changed		
Page 4: [7] Change	Unknown	
Field Code Changed		
Page 4: [7] Change	Unknown	
Field Code Changed		
Page 4: [8] Change	Unknown	
Field Code Changed		
Page 4: [8] Change	Unknown	

Field Code Changed

Page 4: [9] Change	Unknown	
Field Code Changed		
Page 4: [10] Change	Unknown	
Field Code Changed		
Page 4: [10] Change	Unknown	
Field Code Changed		
Page 4: [11] Change	Unknown	
Field Code Changed		
Page 4: [11] Change	Unknown	
Field Code Changed		
Page 4: [12] Change	Unknown	
Field Code Changed		
Page 4: [12] Change	Unknown	
Field Code Changed		
Page 4: [13] Change	Unknown	
Field Code Changed		
Page 4: [13] Change	Unknown	
Field Code Changed		
Page 4: [14] Change	Unknown	
Field Code Changed		
Page 4: [14] Change	Unknown	
Field Code Changed		
Page 4: [15] Change	Unknown	
Field Code Changed		
Page 4: [15] Change	Unknown	
Field Code Changed		
Page 4: [16] Change	Unknown	
Field Code Changed		
Page 4: [16] Change	Unknown	
Field Code Changed		
Page 4: [17] Change	Unknown	
Field Code Changed		

Field Code Changed	
Page 4: [18] Change	Unknown
Field Code Changed	
Page 4: [18] Change	Unknown
Field Code Changed	
Page 4: [19] Change	Unknown
Field Code Changed	
Page 4: [19] Change	Unknown
Field Code Changed	
Page 4: [20] Change	Unknown
Field Code Changed	
Page 4: [20] Change	Unknown
Field Code Changed	
Page 4: [21] Change	Unknown
Field Code Changed	
Page 4: [21] Change	Unknown
Field Code Changed	
Page 4: [22] Change	Unknown
Field Code Changed	
Page 4: [22] Change	Unknown
Field Code Changed	
Page 4: [23] Change	Unknown
Field Code Changed	
Page 4: [23] Change	Unknown
Field Code Changed	
Page 4: [24] Change	Unknown
Field Code Changed	
Page 4: [24] Change	Unknown
Field Code Changed	
Page 4: [25] Change	Unknown
Field Code Changed	

Unknown

Page 4: [25] Change

Page 4: [26] Change	Unknown	
Field Code Changed		
Page 4: [26] Change	Unknown	
Field Code Changed		
Page 4: [27] Change	Unknown	
Field Code Changed		
Page 4: [27] Change	Unknown	
Field Code Changed		
Page 4: [28] Change	Unknown	
Field Code Changed		
Page 4: [28] Change	Unknown	
Field Code Changed		
Page 4: [29] Change	Unknown	
Field Code Changed		
Page 4: [29] Change	Unknown	
Field Code Changed		
Page 4: [30] Change	Unknown	
Field Code Changed		
Page 4: [30] Change	Unknown	
Field Code Changed		
Page 4: [31] Change	Unknown	
Field Code Changed		
Page 4: [31] Change	Unknown	
Field Code Changed		
Page 4: [32] Change	Unknown	
Field Code Changed		
Page 4: [32] Change	Unknown	
Field Code Changed		
Page 4: [33] Change	Unknown	
Field Code Changed		
Page 4: [33] Change	Unknown	
Field Code Changed		

Page 4: [34] Change	Unknown	
Field Code Changed		
Page 4: [35] Change	Unknown	
Field Code Changed		
Page 4: [35] Change	Unknown	
Field Code Changed		
Page 4: [36] Change	Unknown	
Field Code Changed		
Page 4: [36] Change	Unknown	
Field Code Changed		
Page 4: [37] Change	Unknown	
Field Code Changed		
Page 4: [37] Change	Unknown	
Field Code Changed		
Page 4: [38] Change	Unknown	
Field Code Changed		
Page 4: [38] Change	Unknown	
Field Code Changed		
Page 4: [39] Change	Unknown	
Field Code Changed		
Page 4: [39] Change	Unknown	
Field Code Changed		
Page 4: [40] Change	Unknown	
Field Code Changed		
Page 4: [40] Change	Unknown	
Field Code Changed		
Page 4: [41] Change	Unknown	
Field Code Changed		
Page 4: [41] Change	Unknown	
Field Code Changed		
Page 4: [42] Change	Unknown	

Field Code Changed

Field Code Changed		
Page 4: [43] Change	Unknown	
Field Code Changed		
Page 4: [43] Change	Unknown	
Field Code Changed		
Page 4: [44] Change	Unknown	
Field Code Changed		
Page 4: [44] Change	Unknown	
Field Code Changed		
Page 4: [45] Change	Unknown	
Field Code Changed		
Page 4: [45] Change	Unknown	
Field Code Changed		
Page 4: [46] Change	Unknown	
Field Code Changed		
Page 4: [46] Change	Unknown	
Field Code Changed		
Page 4: [47] Change	Unknown	
Field Code Changed		
Page 4: [47] Change	Unknown	
Field Code Changed		
Page 4: [48] Change	Unknown	
Field Code Changed		
Page 4: [48] Change	Unknown	
Field Code Changed		
Page 4: [49] Change	Unknown	
Field Code Changed		
Page 4: [49] Change	Unknown	
Field Code Changed		
Page 4: [50] Change	Unknown	
Field Code Changed		
Dama A. [E0] Channe	Unknown	

Unknown

Page 4: [50] Change

Page 4: [51] Change Field Code Changed

Unknown

Page 4: [51] Change

Unknown

Field Code Changed

Page 6: [52] Formattedfitzgeri4/8/2010 9:47:00 AMTOC 3,toc3, Don't adjust space between Latin and Asian text, Tab stops:72 pt, Left + Not at 24 pt

Page 6: [53] Formatted	fitzgeri	4/8/2010 9:47:00 AM
TOC 2 too2 Depth adjust space k	aturan Latin and Asian taxt. Tab stans	40 mt off Not at 04 mt

TOC 2,toc2, Don't adjust space between Latin and Asian text, Tab stops: 48 pt, Left + Not at 24 pt