



1 PKCS #11 Mechanisms









The following tables are based on PKCS #11 mechanisms specifications version 2.40 and version 3.0.

All mechanism-function-combinations defined by the standard and supported by the SecurityServer are marked with a  character. All mechanism-function-combinations defined by the standard but not supported by the SecurityServer are marked with a  character. Empty mechanism-function-combinations are not defined by the standard.

Mechanisms marked with ^h after the mechanism name were defined in earlier versions than PKCS #11 3.0 but are no longer in general use and have been moved to the Historical Mechanisms Specification. It is not recommended to use them in new applications. Beside these historical mechanisms explicitly mentioned in the tables below, no other historical mechanisms (for example, BATON, CAST, CDMF, FASTHASH, FORTEZZA timestamp, IDEA, JUNIPER, PBE, RIPEMD-128, SKIPJACK) are supported.

All Vendor Defined Mechanisms provided by the SecurityServer are listed in a separate table.

Supported mechanisms are supported equally on all CryptoServer hardware platforms.

Mechanisms	Function						
	Encrypt & Decrypt	Sign & Verify	SR & VR ¹	Digest	Gen. Key (Pair)	Wrap & Unwrap	Derive
PKCS #11 2.40							
CKM_RSA_PKCS_KEY_PAIR_GEN							
CKM_RSA_X9_31_KEY_PAIR_GEN							
CKM_RSA_PKCS	 ²	 ²					
CKM_RSA_PKCS_OAEP	 ²						

CKM_RSA_PKCS_PSS		✓ ²				
CKM_RSA_9796		✗	✗			
CKM_RSA_X_509	✓ ²	✓ ²	✓		✓	
CKM_RSA_X9_31		✓ ²				
CKM_MD2_RSA_PKCS		✗				
CKM_MD5_RSA_PKCS		✓				
CKM_SHA1_RSA_PKCS		✓				
CKM_SHA224_RSA_PKCS		✓				
CKM_SHA256_RSA_PKCS		✓				
CKM_SHA384_RSA_PKCS		✓				
CKM_SHA512_RSA_PKCS		✓				
CKM_RIPEMD128_RSA_PKCS		✗				
CKM_RIPEMD160_RSA_PKCS		✓				
CKM_SHA1_RSA_PKCS_PSS		✓				
CKM_SHA224_RSA_PKCS_PSS		✓				
CKM_SHA256_RSA_PKCS_PSS		✓				
CKM_SHA384_RSA_PKCS_PSS		✓				
CKM_SHA512_RSA_PKCS_PSS		✓				
CKM_SHA1_RSA_X9_31		✓				
CKM_RSA_PKCS_TPM_1_1	✗				✗	
CKM_RSA_PKCS_OAEP_TPM_1_1	✗				✗	
CKM_RSA_AES_KEY_WRAP					✗	

CKM_DSA_KEY_PAIR_GEN					✓		
CKM_DSA_PARAMETER_GEN					✓		
CKM_DSA_PROBABLISTIC_PARAMETER_GEN					✗		
CKM_DSA_SHAW_TAYLOR_PARAMETER_GEN					✗		
CKM_DSA_FIPS_G_GEN					✗		
CKM_DSA		✓ ²					
CKM_DSA_SHA1		✓					
CKM_DSA_SHA224		✓					
CKM_DSA_SHA256		✓					
CKM_DSA_SHA384		✓					
CKM_DSA_SHA512		✓					
CKM_EC_KEY_PAIR_GEN (CKM_ECDSA_KEY_PAIR_GEN)					✓		
CKM_ECDSA		✓ ²					
CKM_ECDSA_SHA1		✓					
CKM_ECDSA_SHA224		✓					
CKM_ECDSA_SHA256		✓					
CKM_ECDSA_SHA384		✓					
CKM_ECDSA_SHA512		✓					
CKM_ECDH1_DERIVE							✓
CKM_ECDH1_COFACTOR_DERIVE							✓

CKM_ECMQV_DERIVE							—
CKM_ECDH_AES_KEY_WRAP						—	
CKM_DH_PKCS_KEY_PAIR_GEN					✓		
CKM_DH_PKCS_PARAMETER_GEN					✓		
CKM_DH_PKCS_DERIVE							✓
CKM_X9_42_DH_KEY_PAIR_GEN					✓		
CKM_X9_42_DH_PARAMETER_GEN					✓		
CKM_X9_42_DH_DERIVE							✓
CKM_X9_42_DH_HYBRID_DERIVE							—
CKM_X9_42_MQV_DERIVE					✓		—
CKM_GENERIC_SECRET_KEY_GEN					✓		
CKM_AES_KEY_GEN					✓		
CKM_AES_ECB	✓					✓	
CKM_AES_CBC	✓					✓	
CKM_AES_CBC_PAD	✓					✓	
CKM_AES_MAC_GENERAL		✓					
CKM_AES_MAC		✓					
CKM_AES_OFB	✓					✓	
CKM_AES_CFB64	—					—	
CKM_AES_CFB8	—					—	
CKM_AES_CFB128	—					—	
CKM_AES_CFB1	—					—	
CKM_AES_XCBC_MAC		—					

CKM_AES_XCBC_MAC_96		—					
CKM_AES_CTR	✓					—	
CKM_AES_CTS	—					—	
CKM_AES_GCM	✓					✓	
CKM_AES_CCM	✓					✓	
CKM_AES_GMAC		✓					
CKM_AES_CMAC_GENERAL		—					
CKM_AES_CMAC		✓					
CKM_AES_KEY_WRAP	✓ ²					✓	
CKM_AES_KEY_WRAP_PAD	✓ ²					✓	
CKM_DES_KEY_GEN ^h					✓		
CKM_DES_ECB ^h	✓					✓	
CKM_DES_CBC ^h	✓					✓	
CKM_DES_CBC_PAD ^h	✓					✓	
CKM_DES_MAC_GENERAL ^h		✓					
CKM_DES_MAC ^h		✓					
CKM_DES2_KEY_GEN					✓		
CKM_DES3_KEY_GEN					✓		
CKM_DES3_ECB	✓					✓	
CKM_DES3_CBC	✓					✓	
CKM_DES3_CBC_PAD	✓					✓	

CKM_DES3_MAC_GENERAL		✓					
CKM_DES3_MAC		✓					
CKM_DES_OFB64	—						
CKM_DES_OFB8	—						
CKM_DES_CFB64	—						
CKM_DES_CFB8	—						
CKM_DES3_CMAC_GENERAL		—					
CKM_DES3_CMAC		—					
CKM_DES_ECB_ENCRYPT_DATA							✓
CKM_DES_CBC_ENCRYPT_DATA							✓
CKM_DES3_ECB_ENCRYPT_DATA							✓
CKM_DES3_CBC_ENCRYPT_DATA							✓
CKM_AES_ECB_ENCRYPT_DATA							✓
CKM_AES_CBC_ENCRYPT_DATA							✓
CKM_MD5 ^h				✓			
CKM_MD5_HMAC_GENERAL ^h		✓					
CKM_MD5_HMAC ^h		✓					
CKM_MD5_KEY_DERIVATION ^h							✓
CKM_SHA_1				✓			
CKM_SHA_1_HMAC_GENERAL		✓					
CKM_SHA_1_HMAC		✓					
CKM_SHA1_KEY_DERIVATION							✓

CKM_SHA224				✓			
CKM_SHA224_HMAC		✓					
CKM_SHA224_HMAC_GENERAL		✓					
CKM_SHA224_KEY_DERIVATION							✓
CKM_SHA256				✓			
CKM_SHA256_HMAC_GENERAL		✓					
CKM_SHA256_HMAC		✓					
CKM_SHA256_KEY_DERIVATION							✓
CKM_SHA384				✓			
CKM_SHA384_HMAC_GENERAL		✓					
CKM_SHA384_HMAC		✓					
CKM_SHA384_KEY_DERIVATION							✓
CKM_SHA512				✓			
CKM_SHA512_HMAC_GENERAL		✓					
CKM_SHA512_HMAC		✓					
CKM_SHA512_KEY_DERIVATION							✓
CKM_SHA512_224				✗			
CKM_SHA512_224_HMAC_GENERAL		✗					
CKM_SHA512_224_HMAC		✗					
CKM_SHA512_224_KEY_DERIVATION							✗
CKM_SHA512_256				✗			
CKM_SHA512_256_HMAC_GENERAL		✗					
CKM_SHA512_256_HMAC		✗					

CKM_SHA512_256_KEY_DERIVATION							–
CKM_SHA512_T				–			
CKM_SHA512_T_HMAC_GENERAL		–					
CKM_SHA512_T_HMAC		–					
CKM_SHA512_T_KEY_DERIVATION							–
CKM_RIPEMD160 ^h				–			
CKM_RIPEMD160_HMAC_GENERAL ^h		–					
CKM_RIPEMD160_HMAC ^h		–					
CKM_PBE_SHA1_DES3_EDE_CBC					–		
CKM_PBE_SHA1_DES2_EDE_CBC					–		
CKM_PBA_SHA1_WITH_SHA1_HMAC					–		
CKM_PKCS5_PBKD2					–		
CKM_SSL3_PRE_MASTER_KEY_GEN					–		
CKM_SSL3_MASTER_KEY_DERIVE							–
CKM_SSL3_MASTER_KEY_DERIVE_DH							–
CKM_SSL3_KEY_AND_MAC_DERIVE							–
CKM_SSL3_MD5_MAC		–					
CKM_SSL3_SHA1_MAC		–					
CKM_TLS_MASTER_KEY_DERIVE							–
CKM_TLS_MASTER_KEY_DERIVE_DH							–
CKM_TLS_KEY_AND_MAC_DERIVE							–
CKM_TLS_PRF							–

CKM_TLS_KDF							—
CKM_TLS_MAC		—					
CKM_TLS12_MASTER_KEY_DERIVE							—
CKM_TLS12_MASTER_KEY_DERIVE_DH							—
CKM_TLS12_KEY_AND_MAC_DERIVE							—
CKM_TLS12_KEY_SAFE_DERIVE							—
CKM_TLS12_KDF							—
CKM_TLS12_MAC		—					
CKM_WTLS_PRE_MASTER_KEY_GEN				—			
CKM_WTLS_MASTER_KEY_DERIVE							—
CKM_WTLS_MASTER_KEY_DERIVE_DH_ECC							—
CKM_WTLS_SERVER_KEY_AND_MAC_DERIVE							—
CKM_WTLS_CLIENT_KEY_AND_MAC_DERIVE							—
CKM_WTLS_PRF							—
CKM_CONCATENATE_BASE_AND_KEY							✓
CKM_CONCATENATE_BASE_AND_DATA							✓
CKM_CONCATENATE_DATA_AND_BASE							✓
CKM_XOR_BASE_AND_DATA							✓
CKM_EXTRACT_KEY_FROM_KEY							✓
CKM_CMS_SIG		—	—				
CKM_BLOWFISH_CBC	—					—	
CKM_BLOWFISH_CBC_PAD	—					—	

CKM_BLOWFISH_KEY_GEN					—		
CKM_TWOFISH_CBC	—					—	
CKM_TWOFISH_CBC_PAD	—					—	
CKM_TWOFISH_KEY_GEN					—		
CKM_CAMELLIA_KEY_GEN					—		
CKM_CAMELLIA_ECB	—					—	
CKM_CAMELLIA_CBC	—					—	
CKM_CAMELLIA_CBC_PAD	—					—	
CKM_CAMELLIA_CTR	—					—	
CKM_CAMELLIA_MAC_GENERAL		—					
CKM_CAMELLIA_MAC		—					
CKM_CAMELLIA_ECB_ENCRYPT_DATA							—
CKM_CAMELLIA_CBC_ENCRYPT_DATA							—
CKM_ARIA_KEY_GEN					—		
CKM_ARIA_ECB	—					—	
CKM_ARIA_CBC	—					—	
CKM_ARIA_CBC_PAD	—					—	
CKM_ARIA_MAC_GENERAL		—					
CKM_ARIA_MAC		—					
CKM_ARIA_ECB_ENCRYPT_DATA							—
CKM_ARIA_CBC_ENCRYPT_DATA							—
CKM_SEED_KEY_GEN					—		
CKM_SEED_ECB			—				

CKM_SEED_CBC			–				
CKM_SEED_CBC_PAD	–					–	
CKM_SEED_MAC_GENERAL			–				
CKM_SEED_MAC				–			
CKM_SEED_ECB_ENCRYPT_DATA							–
CKM_SEED_CBC_ENCRYPT_DATA							–
CKM_SECURID_KEY_GEN					–		
CKM_SECURID		–					
CKM_HOTP_KEY_GEN					–		
CKM_HOTP		–					
CKM_ACTI_KEY_GEN					–		
CKM_ACTI		–					
CKM_KIP_DERIVE							–
CKM_KIP_WRAP						–	
CKM_KIP_MAC		–					
CKM_GOST28147_KEY_GEN					–		
CKM_GOST28147_ECB	–					–	
CKM_GOST28147	–					–	
CKM_GOST28147_MAC		–					
CKM_GOST28147_KEY_WRAP						–	
CKM_GOSTR3411				–			
CKM_GOSTR3411_HMAC		–					
CKM_GOSTR3410_KEY_PAIR_GEN					–		

CKM_GOSTR3410		—					
CKM_GOSTR3410_WITH_GOSTR3411		—					
CKM_GOSTR3410_KEY_WRAP						—	
CKM_GOSTR3410_DERIVECKM_ECDSA_SHA3_384							—
PKCS #11 3.0							
CKM_SHA3_384_RSA_PKCS_PSS		✓					
CKM_SHA3_512_RSA_PKCS_PSS		✓					
CKM_DSA_SHA3_224		✓					
CKM_DSA_SHA3_256		✓					
CKM_DSA_SHA3_384		✓					
CKM_DSA_SHA3_512		✓					
CKM_EC_KEY_PAIR_GEN_W_EXTRA_BITS					—		
CKM_EC_EDWARDS_KEY_PAIR_GEN					✓		
CKM_EC_MONTGOMERY_KEY_PAIR_GEN					—		
CKM_ECDSA_SHA3_224		✓					
CKM_ECDSA_SHA3_256		✓					
CKM_ECDSA_SHA3_384		✓					
CKM_ECDSA_SHA3_512		✓					
CKM_EDDSA		✓					
CKM_XEDDSA		—					
CKM_X3DH_INITIALIZE							—
CKM_X3DH_RESPOND							—

CKM_X2RATCHET_INITIALIZE							—
CKM_X2RATCHET_RESPOND							—
CKM_X2RATCHET_ENCRYPT		—				—	
CKM_X2RATCHET_DECRYPT		—				—	
CKM_AES_XTS		—				—	
CKM_AES_XTS_KEY_GEN					—		
CKM_AES_KEY_WRAP_KWP	✓ ²					✓	
CKM_SHA_1_KEY_GEN					—		
CKM_SHA224_KEY_GEN					—		
CKM_SHA256_KEY_GEN					—		
CKM_SHA384_KEY_GEN					—		
CKM_SHA512_KEY_GEN					—		
CKM_SHA512_224_KEY_GEN					—		
CKM_SHA512_256_KEY_GEN					—		
CKM_SHA512_T_KEY_GEN					—		
CKM_SHA3_224				✓			
CKM_SHA3_224_HMAC		✓					
CKM_SHA3_224_HMAC_GENERAL		✓					
CKM_SHA3_224_KEY_DERIVATION							✓
CKM_SHA3_224_KEY_GEN					—		
CKM_SHA3_256				✓			
CKM_SHA3_256_HMAC		✓					
CKM_SHA3_256_HMAC_GENERAL		✓					

CKM_SHA3_256_KEY_DERIVATION							✓
CKM_SHA3_256_KEY_GEN					–		
CKM_SHA3_384				✓			
CKM_SHA3_384_HMAC		✓					
CKM_SHA3_384_HMAC_GENERAL		✓					
CKM_SHA3_384_KEY_DERIVATION							✓
CKM_SHA3_384_KEY_GEN					–		
CKM_SHA3_512				✓			
CKM_SHA3_512_HMAC		✓					
CKM_SHA3_512_HMAC_GENERAL		✓					
CKM_SHA3_512_KEY_DERIVATION							✓
CKM_SHA3_512_KEY_GEN					–		
CKM_SHAKE_128_KEY_DERIVATION							–
CKM_SHAKE_256_KEY_DERIVATION							–
CKM_BLAKE2B_160				–			
CKM_BLAKE2B_160_HMAC		–					
CKM_BLAKE2B_160_HMAC_GENERAL		–					
CKM_BLAKE2B_160_KEY_DERIVE							–
CKM_BLAKE2B_160_KEY_GEN					–		
CKM_BLAKE2B_256				–			
CKM_BLAKE2B_256_HMAC		–					
CKM_BLAKE2B_256_HMAC_GENERAL		–					
CKM_BLAKE2B_256_KEY_DERIVE							–

CKM_BLAKE2B_256_KEY_GEN					—		
CKM_BLAKE2B_384				—			
CKM_BLAKE2B_384_HMAC		—					
CKM_BLAKE2B_384_HMAC_GENERAL		—					
CKM_BLAKE2B_384_KEY_DERIVE							—
CKM_BLAKE2B_384_KEY_GEN					—		
CKM_BLAKE2B_512				—			
CKM_BLAKE2B_512_HMAC		—					
CKM_BLAKE2B_512_HMAC_GENERAL		—					
CKM_BLAKE2B_512_KEY_DERIVE							—
CKM_BLAKE2B_512_KEY_GEN					—		
CKM_SP800_108_COUNTER_KDF							—
CKM_SP800_108_FEEDBACK_KDF							—
CKM_SP800_108_DOUBLE_PIPELINE_KDF							—
CKM_CHACHA20_KEY_GEN					—		
CKM_CHACHA20	—					—	
CKM_SALSA20_KEY_GEN					—		
CKM_SALSA20	—					—	
CKM_POLY1305_KEY_GEN					—		
CKM_POLY1305		—					
CKM_CHACHA20_POLY1305	—						
CKM_SALSA20_POLY1305	—						
CKM_HKDF_DERIVE							—

SecurityServer 6.0.0

PKCS #11 R3 Mechanisms, Functions and Key Types

CKM_HKDF_DATA							—
CKM_HKDF_KEY_GEN					—		
CKM_NULL	—	—	—	—		—	—

Vendor Defined Mechanisms

Mechanisms	Function						
	Encrypt & Decrypt	Sign & Verify	SR & VR ¹	Digest	Gen. Key (Pair)	Wrap & Unwrap	Derive
CKM_ECDSA_RIPEMD160		✓					
CKM_DSA_RIPEMD160		✓					
CKM_DES3_RETAIL_MAC		✓					
CKM_RSA_PKCS_MULTI		✓ ^{4,5}					
CKM_RSA_X_509_MULTI		✓ ^{4,5}					
CKM_ECDSA_MULTI		✓ ^{4,5}					
CKM_DES_CBC_WRAP						✓	
CKM_AES_CBC_WRAP						✓	
CKM_ECKA		✓ ⁴					
CKM_ECDSA_ECIES	✓ ²						
CKM_ECDSA_SHA256_DCC		✓					
CKM_UTIMACO_SM2		✓					
CKM_UTIMACO_SM2_SHA256		✓					

SecurityServer 6.0.0

PKCS #11 R3 Mechanisms, Functions and Key Types

Mechanisms	Function						
	Encrypt & Decrypt	Sign & Verify	SR & VR ¹	Digest	Gen. Key (Pair)	Wrap & Unwrap	Derive
CKM_UTIMACO_SM2_SHA384		✓					
CKM_UTIMACO_SM2_SHA512		✓					
CKM_UTIMACO_SM2_SHA3_256		✓					
CKM_UTIMACO_SM2_SHA3_384		✓					
CKM_UTIMACO_SM2_SHA3_512		✓					
CKM_UTIMACO_SM2_SM3		✓					
CKM_UTIMACO_SM2_KEY_PAIR_GEN					✓		
CKM_UTIMACO_SM3				✓			
CKM_UTIMACO_SM4_KEY_GEN					✓		
CKM_UTIMACO_SM4_ECB	✓						
CKM_UTIMACO_SM4_ECB_PAD	✓						
CKM_UTIMACO_SM4_CBC	✓						
CKM_UTIMACO_SM4_CBC_PAD	✓						
CKM_UTIMACO_SM4_CFB	✓						
CKM_UTIMACO_SM4_OFB	✓						
CKM_UTIMACO_SM4_CTR	✓						
CKM_UTIMACO_SM4_GCM	✓ ^{2,6}						
CKM_UTIMACO_SM4_CCM	✓ ^{2,6}						
CKM_UTIMACO_SM4_GMAC	✓						

SR = SignRecover

VR = VerifyRecover

2 Single-part operations only



3 Mechanism can only be used for wrapping, not unwrapping.















4 Single-part sign operations only

5 Mechanism can only be used for signing, not for verification.

6 The length of the data to be encrypted must not be zero.

2 PKCS #11 Functions

The following table is based on PKCS #11 base specifications version 2.40 and 3.0. It lists all functions as defined by the standard. Functions supported by the SecurityServer are marked with a  character. Functions not supported by the SecurityServer are implemented as function stub which returns CKR_FUNCTION_NOT_SUPPORTED; they are marked with a  character. Functions introduced with PKCS #11 3.0 specification are marked with "(3.0)" behind the function name.

<i>Function</i>	<i>Description</i>	<i>Supported</i>
General purpose functions		
C_Initialize	initializes Cryptoki	
C_Finalize	clean up miscellaneous Cryptoki associated resources	
C_GetInfo	obtains general information about Cryptoki	
C_GetFunctionList	obtains entry points of Cryptoki library functions	
C_GetInterfaceList (3.0)	obtains entry points of Cryptoki library functions	
C_GetInterface (3.0)	obtains entry points of Cryptoki library functions	
Slot and token management functions		
C_GetSlotList	obtains a list of slots in the system	
C_GetSlotInfo	obtains information about a particular slot	
C_GetTokenInfo	obtains information about a particular token	
C_WaitForSlotEvent	waits for a slot event (token insertion, removal, etc.) to occur	
C_GetMechanismList	obtains a list of mechanisms supported by a token	
C_GetMechanismInfo	obtains information about a particular mechanism	
C_InitToken	initializes a token	
C_InitPIN	initializes the normal user's PIN	

SecurityServer 6.0.0

PKCS #11 R3 Mechanisms, Functions and Key Types

<i>Function</i>	<i>Description</i>	<i>Supported</i>
C_SetPIN	modifies the PIN of the current user	✓
Session management functions		
C_OpenSession		✓
C_CloseSession		✓
C_CloseAllSessions		✓
C_GetSessionInfo		✓
C_SessionCancel (3.0)		—
C_GetOperationState		—
C_SetOperationState		—
C_Login		✓
C_LoginUser	logs a user into a token	✓
C_Logout	logs out from a token	✓
Object management functions		
C_CreateObject	creates an object	✓
C_CopyObject	creates a copy of an object	✓
C_DestroyObject	destroys an object	✓
C_GetObjectSize	obtains the size of an object in bytes	✓
C_GetAttributeValue	obtains an attribute value of an object	✓
C_SetAttributeValue	modifies an attribute value of an object	✓
C_FindObjectsInit	initializes an object search operation	✓
C_FindObjects	continues an object search operation	✓
C_FindObjectsFinal	finishes an object search operation	✓

SecurityServer 6.0.0

PKCS #11 R3 Mechanisms, Functions and Key Types

<i>Function</i>	<i>Description</i>	<i>Supported</i>
Encryption functions		
C_EncryptInit	initializes an encryption operation	✓
C_Encrypt	encrypts single-part data	✓
C_EncryptUpdate	continues a multiple-part encryption operation	✓
C_EncryptFinal	finishes a multiple-part encryption operation	✓
Message-based encryption functions (3.0)		
C_MessageEncryptInit (3.0)	prepares a session for message-based encryption	✓
C_EncryptMessage (3.0)	encrypts a message in a single part	✓
C_EncryptMessageBegin (3.0)	begins a multiple-part encryption operation	✓
C_EncryptMessageNext (3.0)	continues a multiple-part encryption operation	✓
C_MessageEncryptFinal (3.0)	finishes a multiple-part encryption operation	✓
Decryption functions		
C_DecryptInit	initializes a decryption operation	✓
C_Decrypt	decrypts single-part encrypted data	✓
C_DecryptUpdate	continues a multiple-part decryption operation	✓
C_DecryptFinal finishes	finishes a multiple-part decryption operation	✓
Message-based decryption functions (3.0)		
C_MessageDecryptInit (3.0)	prepares a session for message-based decryption	✓
C_DecryptMessage (3.0)	decrypts a message in a single part	✓
C_DecryptMessageBegin (3.0)	begins a multiple-part decryption operation	✓
C_DecryptMessageNext (3.0)	continues a multiple-part decryption operation	✓
C_MessageDecryptFinal (3.0)	finishes a multiple-part decryption operation	✓

SecurityServer 6.0.0

PKCS #11 R3 Mechanisms, Functions and Key Types

<i>Function</i>	<i>Description</i>	<i>Supported</i>
Message digesting functions		
C_DigestInit	initializes a message-digesting operation	✓
C_Digest	digests single-part data	✓
C_DigestUpdate	continues a multiple-part digesting operation	✓
C_DigestKey	digests a key	✓
C_DigestFinal	finishes a multiple-part digesting operation	✓
Signing and MACing functions		
C_SignInit	initializes a signature operation	✓
C_Sign	signs single-part data	✓
C_SignUpdate	continues a multiple-part signature	✓
C_SignFinal	finishes a multiple-part signature operation	✓
C_SignRecoverInit	initializes a signature operation, where the data can be recovered from the signature	✓
C_SignRecover	signs single-part data, where the data can be recovered from the signature	✓
Message-based signing and MACing functions (3.0)		
C_MessageSignInit (3.0)	prepares a session for message-based signing	—
C_SignMessage (3.0)	signs a message in a single part	—
C_SignMessageBegin (3.0)	begins a multiple-part signing operation	—
C_SignMessageNext (3.0)	continues a multiple-part signing operation	—
C_MessageSignFinal (3.0)	finishes a multiple-part signing operation	—
Functions for verifying signatures and MACs		
C_VerifyInit	initializes a verification operation	✓

SecurityServer 6.0.0

PKCS #11 R3 Mechanisms, Functions and Key Types

<i>Function</i>	<i>Description</i>	<i>Supported</i>
C_Verify	verifies a signature on single-part data and MACs	✓
C_VerifyUpdate	continues a multiple-part verification operation	✓
C_VerifyFinal	finishes a multiple-part verification operation	✓
C_VerifyRecoverInit	initializes a verification operation where the data is recovered from the signature	✓
C_VerifyRecover	verifies a signature on single-part data, where the data is recovered from the signature	✓
Message-based functions for verifying signatures and MACs (3.0)		
C_MessageVerifyInit (3.0)	prepares a session for message-based verifying	—
C_VerifyMessage (3.0)	verifies a signature in a single part	—
C_VerifyMessageBegin (3.0)	begins a multiple-part verifying operation	—
C_VerifyMessageNext (3.0)	continues a multiple-part verifying operation	—
C_MessageVerifyFinal (3.0)	finishes a multiple-part verifying operation	—
Dual-function cryptographic functions		
C_DigestEncryptUpdate	continues simultaneous multiple-part digesting and encryption operations	✓
C_DecryptDigestUpdate	continues simultaneous multiple-part decryption and digesting operations	✓
C_SignEncryptUpdate	continues simultaneous multiple-part signature and encryption operations	✓
C_DecryptVerifyUpdate	continues simultaneous multiple-part decryption and verification operations	✓
Key management functions		
C_GenerateKey	generates a secret key	✓

SecurityServer 6.0.0

PKCS #11 R3 Mechanisms, Functions and Key Types

<i>Function</i>	<i>Description</i>	<i>Supported</i>
C_GenerateKeyPair	generates a public-key/private-key pair	✓
C_WrapKey	wraps (encrypts) a key	✓
C_UnwrapKey	unwraps (decrypts) a key	✓
C_DeriveKey	derives a key from a base key	✓
Random number generation functions		
C_SeedRandom	mixes in additional seed material to the random number generator	✓
C_GenerateRandom	generates random data	✓
Parallel function management functions		
C_GetFunctionStatus	legacy function which always returns CKR_FUNCTION_NOT_PARALLEL	—
C_CancelFunction	legacy function which always returns CKR_FUNCTION_NOT_PARALLEL	—

3 PKCS #11 Key Types

Key Type	Description	Supported
CKK_RSA	RSA public or private key	✓
CKK_DSA	DSA public or private key	✓
CKK_EC	EC public or private key	✓
CKK_EC_MONTGOMERY	Montgomery EC public or private key	✗
CKK_EC_EDWARDS	Edwards EC public or private key	✓
CKK_DH	Diffie-Hellmann public or private key	✓
CKK_X9_42_DH	X9.42 Diffie-Hellman public or private key	✓
CKK_X2RATCHET	Double Ratchet secret key	✗
CKK_GENERIC_SECRET	Generic secret key	✓
CKK_AES	AES secret key	✓
CKK_AES_XTS	Double-length AES secret key	✗
CKK_DES ^h	DES secret key	✓
CKK_DES2	Double-length DES secret key	✓
CKK_DES3	Triple-length DES secret key. Triple DES (TDES) is blocked for FIPS on u.trust Anchor.	✓
CKK_SHA1_HMAC	Secret key to be used with the SHA-1-HMAC mechanisms	✗
CKK_SHA224_HMAC	Secret key to be used with the SHA-224-HMAC mechanisms	✗
CKK_SHA256_HMAC	Secret key to be used with the SHA-256-HMAC mechanisms	✗
CKK_SHA384_HMAC	Secret key to be used with the SHA-384-HMAC mechanisms	✗
CKK_SHA512_HMAC	Secret key to be used with the SHA-512-HMAC mechanisms	✗

Key Type	Description	Supported
CKK_SHA512_224_HMAC	Secret key to be used with the SHA-512/224 mechanisms	—
CKK_SHA512_256_HMAC	Secret key to be used with the SHA-512/256 mechanisms	—
CKK_SHA512_T_HMAC	Secret key to be used with the SHA-512/t mechanisms	—
CKK_SHA3_224_HMAC	Secret key to be used with the SHA3-224-HMAC mechanisms	—
CKK_SHA3_256_HMAC	Secret key to be used with the SHA3-256-HMAC mechanisms	—
CKK_SHA3_384_HMAC	Secret key to be used with the SHA3-384-HMAC mechanisms	—
CKK_SHA3_512_HMAC	Secret key to be used with the SHA3-512-HMAC mechanisms	—
CKK_BLAKE2B_160_HMAC	Secret key to be used with the BLAKE2B-160-HMAC mechanisms	—
CKK_BLAKE2B_256_HMAC	Secret key to be used with the BLAKE2B-256-HMAC mechanisms	—
CKK_BLAKE2B_384_HMAC	Secret key to be used with the BLAKE2B-384-HMAC mechanisms	—
CKK_BLAKE2B_512_HMAC	Secret key to be used with the BLAKE2B-512-HMAC mechanisms	—
CKK_BLOWFISH	Blowfish secret key	—
CKK_TWOFISH	Twofish secret key	—
CKK_CAMELLIA	Camellia secret key	—
CKK_ARIA	ARIA secret key	—
CKK_SEED	SEED secret key	—
CKK_SECURID	RSA SecurID secret key	—
CKK_HOTP	OATH HOTP secret key	—
CKK_ACTI	ActivIdentity ACTI secret key	—
CKK_GOST28147	GOST 28147-89 secret key	—
CKK_GOSTR3411	GOST R 34.11-94 domain parameters	—
CKK_GOSTR3410	GOST R 34.10-2001 public or private key	—

SecurityServer 6.0.0

PKCS #11 R3 Mechanisms, Functions and Key Types

<i>Key Type</i>	<i>Description</i>	<i>Supported</i>
CKK_CHACHA20	ChaCha20 secret key	—
CKK_SALSA20	Salsa20 secret key	—
CKK_POLY1305	Poly1305 secret key	—
CKK_HKDF	HMAC-based KDF (HKDF) secret key	—