GLOBALPLATFORM®

THE STANDARD FOR MANAGING APPLICATIONS ON SECURE CHIP TECHNOLOGY

GlobalPlatform Card

Opacity Secure Channel

Card Specification v2.3 – Amendment G

Version 1.0

Public Release

October 2016

Document Reference: GPC_SPE_106

THIS SPECIFICATION OR OTHER WORK PRODUCT IS BEING OFFERED WITHOUT ANY WARRANTY WHATSOEVER, AND IN PARTICULAR, ANY WARRANTY OF NON-INFRINGEMENT IS EXPRESSLY DISCLAIMED. ANY IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT SHALL BE MADE ENTIRELY AT THE IMPLEMENTER'S OWN RISK, AND NEITHER THE COMPANY, NOR ANY OF ITS MEMBERS OR SUBMITTERS, SHALL HAVE ANY LIABILITY WHATSOEVER TO ANY IMPLEMENTER OR THIRD PARTY FOR ANY DAMAGES OF ANY NATURE WHATSOEVER DIRECTLY OR INDIRECTLY ARISING FROM THE IMPLEMENTATION OF THIS SPECIFICATION OR OTHER WORK PRODUCT.

Contents

Figures

Tables

1 Introduction

The purpose of this amendment is to extend the GlobalPlatform Card Specification [GPCS] with new secure channel and key establishment protocols, altogether known as the Opacity Secure Channel establishment method or **Secure Channel Protocol '22'**.

The Opacity Secure Channel establishment method includes:

- Opacity ZKM and Opacity FS protocols as defined in [ANSI 504-1]
- Opacity Blinded protocol as defined in this specification

Configurations will define the protocol modes, the selection of cipher suites and secure messaging formats that are mandatory or optional in specific market segments.

1.1 Audience

This amendment is intended primarily for card manufacturers and developers of applications for GlobalPlatform cards.

It is assumed that the reader is familiar with smart cards and smart card production, and in particular familiar with the GlobalPlatform Card Specification [GPCS].

1.2 IPR Disclaimer

Attention is drawn to the possibility that some of the elements of this GlobalPlatform specification or other work product may be the subject of intellectual property rights (IPR) held by GlobalPlatform members or others. For additional information regarding any such IPR that have been brought to the attention of GlobalPlatform, please visit https://www.globalplatform.org/specificationsipdisclaimers.asp. GlobalPlatform shall not be held responsible for identifying any or all such IPR, and takes no position concerning the possible existence or the evidence, validity, or scope of any such IPR.

1.3 References

Table 1-1: Normative References

Standard / Specification	Description	Ref
GlobalPlatform Card Specification	GlobalPlatform Card Specification v2.3	[GPCS]
GlobalPlatform Card Specification Amendment D	GlobalPlatform Card Technology Secure Channel Protocol '03' Card Specification v2.2 – Amendment D v1.1.1	[GPCS-D]
GlobalPlatform Card Specification ISO Framework	GlobalPlatform Card Technology Card Specification – ISO Framework v1.0	[GPCS-ISO]
FIPS PUB 186-4	Digital Signature Standard (DSS) FIPS PUB 186-4	[FIPS 186-4]
ANSI/INCITS 504-1:2013	INCITS 504-1 – Generic Identity Command Set Part 1: Card Application Command Set	[ANSI 504-1]

Standard / Specification	Description	Ref
ANSI X9.62:2005	Public Key Cryptography for the Financial Services Industry, The Elliptic Curve Digital Signature Algorithm (ECDSA)	[ANSI X9.62]
NIST SP 800-38B	Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, May 2005	[NIST 800-38B]
NIST SP 800-56A Revision 2	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography, Revision 2 May 2013	[NIST 800-56A]
NIST SP 800-56C	Recommendation for Key Derivation through Extraction-then-Expansion. November 2011	[NIST 800-56C]
NIST SP 800-73-4	Interfaces for Personal Identity Verification – May 2015	[NIST 800-73-4]
NIST SP 800-90A	Recommendation for Random Number Generation Using Deterministic Random Bit Generators, January 2012	[NIST 800-90A]
NIST SP 800-108	Recommendation for Key Derivation Using Pseudorandom Functions (Revised), October 2009.	[NIST 800-108]
ISO/IEC 7816-4:2005	Identification cards – Integrated circuit cards – Part 4: Organization, security and commands for interchange	[ISO 7816-4]
ISO/IEC 7816-8:2004	Identification cards – Integrated circuit cards – Part 8: Commands for security operations	[ISO 7816-8]
ISO/IEC 19772/AC1:2014	Information technology – Security techniques – Authenticated encryption [ISO/IEC 19772:2009 with Technical correction]	[ISO 19772]
BSI TR-02102-1	BSI Technische Richtlinie TR-02102-1: Kryptographische Verfahren: Empfehlungen und Schlüssellängen (Cryptographic Methods: Recommendations and Key Lengths) v2015-01	[TR 02102]
BSI TR-03111	BSI Technical Guideline TR-03111: Elliptic Curve Cryptography. Version 2.0	[TR 03111]
RFC 5639	Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation	[RFC 5639]
SM2	Chinese Commercial Cryptography Administration Office: SM2 Digital Signature Algorithm, December 2010, http://www.oscca.gov.cn/UpFile/2010122214822692.pdf	[SM2]
SM3	Chinese Commercial Cryptography Administration Office: SM3 Cryptographic Hash function, December 2010, http://www.oscca.gov.cn/UpFile/20101222141857786.pdf	[SM3]
SMS4	Chinese Commercial Cryptography Administration Office: Block cipher for WLAN products SMS4, September 2014, http://www.oscca.gov.cn/UpFile/2006210164231979 90.pdf	[SMS4]

Table 1-2: Informative References

Document	Description	Ref
An analysis of the EMV channel establishment protocol.	Christina Brzuska, Nigel P. Smart, Bogdan Warinschi, and Gaven J. Watson. An analysis of the EMV channel establishment protocol. ACM Conference on Computer and Communications Security – ACM CCS 2013, 373-386, 2013	[Smart]
NIST SP 800-57 Part 1 Revision 3	Recommendation for Key Management – Part 1: General (Revision 3), July 2013.	[NIST 800-57]
ISO/IEC 7812	Identification cards – Identification of Issuers	[ISO 7812]
ISO 9797-1	Information technology – Security techniques – Message Authentication Codes (MACs) – Part 1: Mechanisms using a block cipher	[ISO 9797-1]
ISO/IEC 14888-3:2006	Information technology – Security techniques – Digital signatures with appendix – Part 3: Discrete logarithm based mechanisms	[ISO 14888]
SM2	Chinese Commercial Cryptography Administration Office: SM2 Digital Signature Algorithm Translation, February 2014: tools.ietf.org/html/draft-shen-sm2-ecdsa-02	[SM2-Eng]
SM3	Chinese Commercial Cryptography Administration Office: SM3 Cryptographic Hash function Translation, February 2014: tools.ietf.org/html/draft-shen-sm3-hash-00	[SM3-Eng]
SMS4	Chinese Commercial Cryptography Administration Office: Block cipher for WLAN products SMS4 Translation, September 2014: http://eprint.iacr.org/2008/329.pdf	[SMS4-Eng]
Advanced Encryption Standard (AES)	Federal Information Processing Standards Publication 197: Specification for the Advanced Encryption Standard (AES) Available at FIPS Pub 197	[FIPS-197]

1.4 Terminology and Definitions

The technical terms used in this amendment are defined in [GPCS].

1.5 Abbreviations and Notations

Table 1-3: Abbreviations and Notations

Abbreviation / Notation	Meaning
AD	Associated Data
AEAD	Authenticated Encryption with Associated Data
AES	Advanced Encryption Standard
ANSI	American National Standards Institute
APDU	Application Protocol Data Unit
BSI	Bundesamt für Sicherheit in der Informationstechnik
CB	Control Byte
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC (AEAD Mode)
CLA	Class byte of the command message
CMAC	Cipher-based MAC ([NIST 800-38B])
CS	Cipher Suite
CT	Cipher Text
CTR-DRBG	A block cipher based PRNG ([NIST 800-90A])
CVC	Card Verifiable Certificates
dn	Distinguished Name
DRBG	Deterministic Random Bit Generator
DSS	Digital Signature Standard
EAX	EAX (AEAD Mode)
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie Hellman
ECDSA	Elliptic Curve Digital Signature Algorithm
EtM	Encrypt then MAC (AEAD mode)
FIPS	Federal Information Processing Standards
FS	Full Secrecy
GCM	Galois Counter Mode (AEAD mode)
GF(p)	Galois Field or Finite Field with p elements – p being a prime
HMAC	Hash-based Message Authentication Code
HMAC-DRBG	Hash-based PRNG
IIN	Issuer Identification Number

Abbreviation / Notation	Meaning
INCITS	InterNational Committee for Information Technology Standards
INS	Instruction byte of command message
KDF	Key Derivation Function
MAC	Message Authentication Code
N/A	Not applicable
NIST	National Institute of Standards and Technology
OCB2	Offset Codebook Mode
OID	Object IDentifier
PRNG	Pseudo Random Number Generator
PT	Plain Text
RFU	Reserved for Future Use
RNG	Random Number Generator
SCP	Secure Channel Protocol
SE	Secure Element
SHA-1	Secure Hash Algorithm 1 (digest size 160 bits)
SHA-*nnn*	Secure Hash Algorithm with digest size *nnn* bits
ShS	Shared Secret
SK	Session Key
TLV	Data structure containing of Tag, Length, Value fields
URI	Uniform Resource Identifier
XOR	Exclusive or
ZKM	Zero Key Management

1.6 Revision History

Table 1-4: Revision History

Date	Version	Description
October 2016	1.0	Public Release

2 Use Cases

The purpose of this specification is to present and specify a compact and efficient method to establish secure channel keys between an entity (Host) and a secure element that can be authenticated using a public/private elliptic curve key pair. The secure channel keys are established in a single request response step, and can be used to protect further communication between the SE and Host (host or terminal).

See [ANSI 504-1] and [NIST 800-73-4] Part II section 4.1 for specific use cases involving Opacity ZKM mode and Opacity FS mode.

The supplementary Secure Channel establishment method presented in this specification, i.e. the Opacity Blinded protocol, allows correct implementations to make the following security claims:

- **One-way entity authentication** – The identity claimed in the response message from the SE can be proven to be bound to the SE secret.
- **One-way messaging privacy** – The identity of the SE is carried in the response but not revealed.
- **One-way message authentication** – The response message from the SE can be proven to be bound to the SE secret.
- **Non-traceability** – The information from two messages cannot be correlated to reveal the message origin, or to reveal the message content.
- **Forward Secrecy** – The compromise of the long term SE secret does not compromise the message privacy or confidentiality of any payload sent in the past.

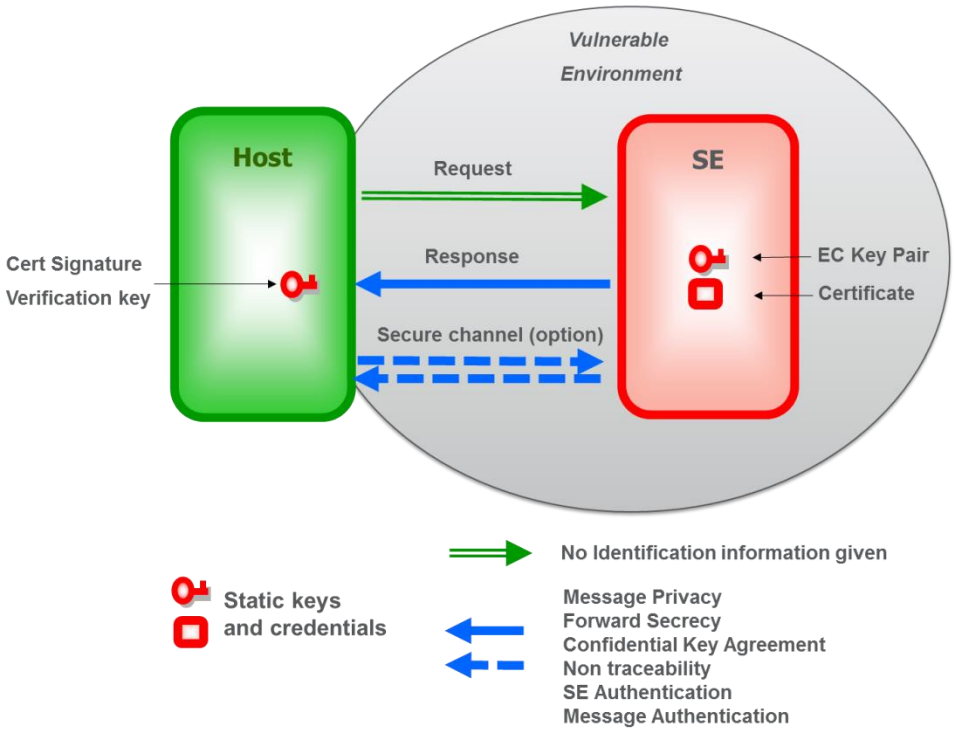
The privacy and non-traceability properties claimed above depend on the initial assumption that the host is not an active attacker, and would not be valid otherwise. Host Authentication may be performed at the application level using the secure messaging resulting from the execution of the protocol.

A formal analysis to prove the above security properties has been established in 'An analysis of the EMV channel establishment protocol' ([Smart]). The proofs on privacy, non-traceability, and forward secrecy assume that a full size blinding factor is being used, i.e., the blinding factor shall have the same number of bits as the curve order.

Using standard-based cryptography in [GPCS]:

- The messaging specification allows a mapping to APDUs or other communication channels.

Figure 2-1: Opacity Blinded Protocol Overview



3 Algorithms

3.1 ECC

3.1.1 EC Curves

Elliptic Curve Cryptography over prime fields GF(p) shall be used for the purpose of this amendment.

Standardized Domain Parameters are recommended to be used:

- Digital Signature Standard (DSS) [FIPS 186-4], recommended by NIST, or
- Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation [RFC 5639], recommended by BSI.

These references specify the EC domain parameters for curves of different key lengths:

P: Prime field specification parameter

A,**B**: EC curve equation parameters

G: Base EC Point

N: EC order parameter

k: EC cofactor parameter

The following table lists recommended curves for different ECC key lengths.

Table 3-1: ECC Key Length and Recommended Curves

Curve Specified in [FIPS 186-4]	Curve Specified in [RFC 5639]	Curve Specified in [SM2]	ECC Key Length (bits)
P-256	brainpoolP256r1 brainpoolP256t1	Fp-256	256
P-384	brainpoolP384r1 brainpoolP384t1	N/A	384
N/A	brainpoolP512r1 brainpoolP512t1	N/A	512
P-521	N/A	N/A	521

3.1.2 Random Number Generators

Recommendations for appropriate random number generators are given in BSI TR-02102 [TR 02102] and NIST SP 800-90A [NIST 800-90A].

The random number generator involves an entropy source and a Pseudo Random Number Generator (PRNG) applied on the source output. Example PRNGs are defined in [NIST 800-90A]:

- CTR-DRBG (Block cipher based PRNG)
- HMAC-DRBG or Hash_DRBG (Hash-based PRNG)

3.1.3 EC Private Key

A private key is a randomly generated number with the size of the EC Key Length in bits. A recommended number generator must be used (see section 3.1.2).

3.1.4 EC Points

Any point Q on the curve with x- and y-coordinates (x,y) shall be encoded either as:

- In [TR 03111] section 3.2.
- An optimized encoding, i.e. the byte string representation of the x-coordinate only. The y-coordinate is chosen as the lowest value obtained by applying the curve equation to the x-coordinate.

The multiplication of any point (x, y) on the curve with an integer number is another point on the curve (x', y').

3.1.5 EC Public Key

The multiplication of the base point G with a private key d is the associated public key.

3.1.6 EC Public Key Blinding

The multiplication of any EC Public Key with a random blinding factor, which size in bits is comprised between n and n/2 where n is the number of bits the EC curve order N.

3.1.7 ECDSA

ECDSA is specified in ANSI X9.62 [ANSI X9.62] standard.

From an input message M, ECDSA produces a signature (r,s).

ECDSA requires a hash function. Unless defined otherwise, the security strength of the hash function used shall meet or exceed the security strength associated with the order of the key according to [FIPS 186-4].

The following table lists the Hash algorithms that shall be used depending on specific ECC key lengths.

Table 3-2: Hash Algorithms for ECDSA

ECC Key Length (in bits)	Hash Algorithm
256-383	SHA-256, SHA-384, SHA-512
384-511	SHA-384, SHA-512
512+	SHA-512

The signature shall be coded according to the signed object format (see section 3.2.4). The ECDSA signature algorithm requires a random value as input. To protect against attacks, a high quality random number generator is required for the entity generating the signature. Recommendations for appropriate random number generators are given in section 3.1.2.

3.2 Certificates

3.2.1 Certificate Types

X.509 v3 and Card Verifiable Certificates (CVC) in section 3.2.4 are the two certificate types recommended in this version of the specification.

3.2.2 Trusted Certificates

Any entity validating that a certificate can be trusted must first trust the corresponding signing certificate.

A validating entity must therefore trust the original signing certificates such as root certificates or CA certificates or at least trust the public key association with the issuer, root, or CA identity. The process allowing an entity to trust an issuer, root, or CA certificate or its public key association is outside the scope of this specification.

3.2.3 Certificate Chains

Certificates for which signing certificates are not already trusted by the validating entity may be presented instead as certificate chains, i.e. a sequence of certificates. Each certificate in the sequence is the signing certificate for the following certificate in the chain. The first certificate must be verifiable by the validating entity.

3.2.4 CVC Format (Informative)

The Card Verifiable Certificate Format used in this specification follows [ISO 7816-8] and [ANSI 504-1] as follows:

Table 3-3: Card Verifiable Certificate Format

Tag	Tag	Tag	Length	Name	Value
'7F21'				Card Verifiable Certificate	
	'5F29'		1	Credential Profile Identifier	'80'
	'42'		8	Issuer Identification Number	IIN: 8 bytes: leftmost 8 bytes of the subjectKeyIdentifier of the CVC signer X.509 certificate. This references the CVC signer key SIGKEYID.
	'5F20'		Var.	Subject Identifier	Actual identity dn or URI associated with public key
	'7F49'		Var.	PublicKey Data Object	Composite object
		'06'	Var.	Algorithm OID	Public Key Curve OID o {1.3.36.3.3.2.8.1.1.7} brainpoolP256r1 o {1.3.36.3.3.2.8.1.1.8} brainpoolP256t1 o {1.3.36.3.3.2.8.1.1.11} brainpoolP384r1 o {1.3.36.3.3.2.8.1.1.12} brainpoolP384t1 o {1.3.36.3.3.2.8.1.1.13} brainpoolP512r1 o {1.3.36.3.3.2.8.1.1.14} brainpoolP512t1 o {1.2.840.10045.3.1.7} secp256r1 o {1.3.132.0.34} secp384r1 o {1.3.132.0.35} secp521r1 This is encoded as DER. For instance '2A8648CE3D030107' for secp256r1
		'86'	Var.	Public Key object	Coded as follows: o '04' || X || Y, where X and Y are the coordinates of the point on the curve

Tag	Tag	Tag	Length	Name	Value

	'5F37'		Var.	DigitalSignature object	DigitalSignature ::= SEQUENCE {
					signatureAlgorithm AlgorithmIdentifier,
					signatureValue ECDSA-Sig-Value
					}
					
					Where:
					AlgorithmIdentifier ::= SEQUENCE {
					algorithm OBJECT IDENTIFIER,
					parameters ANY DEFINED BY algorithm OPTIONAL
					}
					
					Algorithm allowed OID values are:
					o {1.2.840.10045.4.3.2} ECDSA with SHA-256
					o {1.2.840.10045.4.3.3} ECDSA with SHA-384
					o {1.2.840.10045.4.3.4} ECDSA with SHA-512
					And Where:
					ECDSA-Sig-Value ::= SEQUENCE {
					r INTEGER,
					s INTEGER
					}
					Note:
					The signature is computed using the corresponding signing key SIGKEYID, referenced with the IssuerID Data Element.
					EC Domain parameters are derived from tag '06' of PublicKey Data Object.

Tag	Tag	Tag	Length	Name	Value
	'5F4C'		1	Role Identifier	Indicates the type of certificate The format is: b8b7b6b5b4b3b2b1 For root, CA, or issuer certificates: 0 0 x 1 - - - - Key assigned to SE app. 0 0 1 x - - - - Key assigned to client app. - - - - 0 0 0 1 Verification key - - - - 0 0 1 0 Authentication Key Card application root CVC role ID: '12' Client application root CVC role ID: '22' For other certificates: '00' : for card-application key CVC '80' : for card-application administrator key CVC '01': for client application key CVC

3.2.5 Certificate Validation

The following steps are recommended to validate a certificate.

- Verify signature on the certificate (see section 3.1.7).
- Validate the contents of the certificate, at minimum Key usage, Algorithm OID, domain parameters, and Issuer identifier.

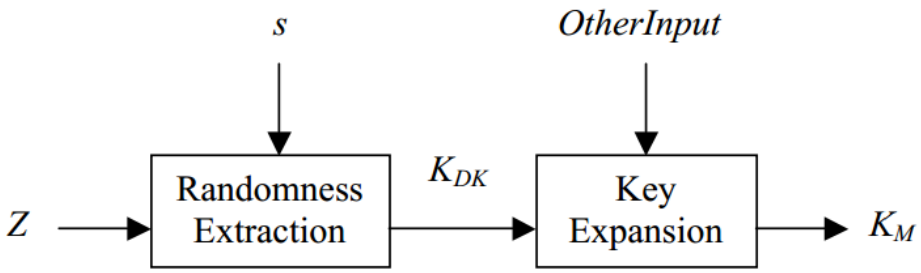
3.3 Key Derivation

Two Key Derivation Functions (KDF) are defined in this specification. A 1-step KDF (KDF1S) using Secure Hash Algorithm defined in [NIST 800-56A] and a 2-step KDF (KDF2S) using AES-CMAC allowed in [NIST 800-56A] and defined in [NIST 800-56C]. KDF2S is proposed as a more efficient method than KDF1S as it only relies on AES primitives with no hash function. KDF1S is chosen as compliant with [ANSI 504-1].

3.3.1 2-Step Key Derivation Function (KDF2S)

This section describes the 2-step Key Derivation Function (KDF2S), also specified in [NIST 800-56C], and makes appropriate choices for Opacity.

Figure 3-1: Extraction-then-Expansion Key Derivation Procedure



Extraction Step

The extraction-then-expansion key derivation procedure begins with a shared secret Z. Z is the x-coordinate of the common EC point computed on both sides.

Z length in bytes is defined according to the following rule:

Len (Z) is equal to the lowest number of bytes that can include a number of bits equal to the EC curve order.

Example:

For P-256 or brainpoolP256r1: len(Z) = 256/8 = 32 bytes

For P-521: len(Z) = 521/8 = 65.125. The lowest number of bytes to include the curve order in bits is 66 bytes.

The Randomness Extraction function is the MAC function (AES-CMAC).

s – Salt, a (public or secret) byte string used as the “key” during the execution of the randomness extraction step. s is the all-zero byte string. The bit length of the all-zero byte string shall equal the length of the AES key used, e.g. for AES 128:

s = '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'

K_{DK} – The output of the randomness extraction step.

When AES-CMAC is used, K_{DK} is a binary string of length 128 bits.

1. K_{DK} = AES-CMAC (s, Z)
2. Zeroize Z.

Expansion Step

This section describes the Key Derivation Function (KDF) common to all session keys regardless of the method used to generate the shared secret.

The expansion step is as specified in NIST SP 800-108 [NIST 800-108] in counter mode (§5.1). It takes as input the result of the extraction step K_{DK}, the total session key data length '*len*', and supplementary data '*OtherInput*' consisting of multiple data elements A_1, .. , A_m, and produces session keys as follows:

OtherInput (A_1, .. , A_m) = [AlgorithmID(SK_1) || .. || AlgorithmID(SK_p) || A_1 || .. || A_m]

len = algoKeyLengthInBits(SK_1) + … + algoKeyLengthInBits(SK_p)

n = len / (hashLengthInBits (KDFHashAlgorithm)) = len/128

K_M = [AES-CMAC (K_{DK}, '00 00 00 01' || OtherInput) || AES-CMAC (K_{DK}, '00 00 00 02' || OtherInput) || … || AES-CMAC (K_{DK}, ('00 00 00 00' + n || OtherInput)]

With:

The concatenated session keys (SK_1 || .. || SK_p) are the (*len*) leftmost bits from K_M.

3.3.2 1-step Key Derivation Function (KDF1S)

This section describes the 1-step Key Derivation Function (KDF1S), which is based on the use of the shared secret previously exchanged using public key cryptography. It is common to all session keys regardless of the method used to generate the shared secret.

The Key Derivation Function is implemented as specified in [NIST 800-56A] §5.8.1. It takes as input a shared secret 'Z', total session key data length '*len*', and supplementary data '*info*', and produces session keys as follows:

The concatenated session keys (**SK_1 || .. || SK_p**) are the (*len*) leftmost bits from *DerivedKeyingMaterial*.

With:

len = algoKeyLengthInBits(**SK_1**) + … + algoKeyLengthInBits(**SK_p**)

DerivedKeyingMaterial = [KDFHashAlgorithm('00 00 00 01' || Z || info) || KDFHashAlgorithm('00 00 00 02' || Z || info) || KDFHashAlgorithm(('00 00 00 00' + n) || Z || info)]

With:

KDFHashAlgorithm = SHA-256 or SHA-384 or SHA-512 (See Table 3-5 below.)

n = len / (hashLengthInBits (KDFHashAlgorithm))

Z = Shared Secret

Info (A_1, .. , A_m) = [**AlgorithmID(SK_1) || .. || AlgorithmID(SK_p)** || A_1|| .. || A_m]

With:

Table 3-4: KDF Configuration Based on Input Curve

ECDH Curve	Z Length in Bytes	AlgorithmID (SK)	SK Length in Bytes
P-256 or Brainpool 256 or SM2 Fp-256	16	AES 128 or SMS4 128 for SM2	16
P-384 or Brainpool 384	24	AES 192	24
P-521 or Brainpool 512	32	AES 256	32

3.4 Authenticated Encryption

3.4.1 Authenticated Encryption with Associated Data (AEAD)

The Authenticated encryption modes defined in section 3.4.3 allow the protection of data for confidentiality and integrity as follows:

AEAD (SK, AD, PT) = AD || CT || MAC

Where:

SK = Session Key

AD = Associated Data. Will not be encrypted.

PT = Plain text. Will be encrypted.

CT = Cipher Text resulting from encryption of PT with SK

MAC = Integrity check value using SK computed over the concatenation of AD (clear text) and CT (cipher text).

3.4.2 Authenticated Encryption without Associated Data (AE)

Authenticated Encryption without Associated Data (AE) is defined as follows:

AE (SK, PT) = AEAD (SK, NULL, PT) = CT || MAC

3.4.3 AEAD Modes

One of the following available AEAD modes is required:

Table 3-5: Available AEAD Modes

Mode	Reference
OCB2	See [ISO 19772] section 6.
CCM	See [ISO 19772] section 8.
EAX	See [ISO 19772] section 9.
Encrypt then MAC (ETM-CTR-CMAC)	See [ISO 19772] section 10. Use AES Counter Mode encryption and AES-CMAC.
GCM	See [ISO 19772] section 11.

3.4.4 Block Cipher

AES is the default Block Cipher Algorithm used for AEAD.

When SM2 Curves are selected, SMS4 must be used instead.

4 Protocol

This chapter specifies the Opacity Blinded protocol. The protocol for Opacity ZKM and Opacity FS are specified in [ANSI 504-1].

4.1 Cipher Suites

The Opacity Blinded protocol uses several cryptographic mechanisms which can be configured in different ways. Such configurations are known as 'Cipher Suites' and can be defined according to the requirements defined in Table 4-1.

Table 4-1: Requirements for Opacity Cipher Suites

Cipher Suites	Opacity-128	Opacity-192	Opacity-256
Channel Strength (bits)	**128**	**192**	**256**
Session keys – (see AEAD mode)	AES128-AEAD or SMS4-128-AEAD for SM2 curve	AES 192-AEAD	AES256-AEAD
SE certificate Signature	ECDSA 256 (*)	ECDSA 384 (*)	ECDSA 521 (*)
SE certificate Signature Hash	SHA-256 (*)	SHA-384 (*)	SHA-512 (*)
Key Agreement Curve	P-256 or brainpoolP256r1 or brainpoolP256t1 or SM2 Fp-256	P-384 or brainpoolP384r1 or brainpoolP384t1	P-521 or brainpoolP512r1 or brainpoolP512t1
KDF	KDF1S or KDF2S	KDF1S or KDF2S	KDF1S or KDF2S
Nonces ([NIST 800-56A])	16 bytes	24 bytes	32 bytes
Authenticated Encryption Mode	See section 3.4.3.		
Secure Messaging	See section 4.5		

(*) or equivalent strength specified in the certificate.

In this specification, Cipher Suites are defined for the Opacity protocol and are assigned a UTF-8 string name according to the following pattern:

```
OPACITY-{bit_strength}-{key_agreement_mode}-{curve_type}-
{key_derivation_function}-{AEAD_mode}-{secure_messaging_mode}
```

Where :

- `Bit_strength`
  - 128, 192, or 256.
- `key_agreement_mode`
  - `ECDHB` : usage of Elliptic Curve Diffie-Hellman Blinded Mode (as described in this document).
- `curve_type`
  - `NIST` means usage of the NIST curve according to the specified bit strength (e.g. P256 for Opacity-128).
  - `BR1` means usage of the Brainpool r1 curve according to the specified bit strength (e.g. brainpoolP256r1 for Opacity-128).
  - `SM2` means usage of the SM2 curve according to the specified bit strength (e.g. SM2 Fp-256 for Opacity-128).
- `key_derivation_function`
  - `KDF1S`: one–step KDF
  - `KDF2S`: two–step KDF
- `AEAD_mode`
  - `CCM`, `GCM`, and `ETM` are the recommended AEAD modes necessary to compute the response of the authentication command. The selection also applies to the AEAD secure messaging mode when it is selected.
- `secure_messaging_mode`
  - `AEAD` means usage of AEAD secure messaging. The AEAD mode used to protect the response to the authentication command message and for the AEAD secure messaging shall be the same.
  - `SCP03` means usage of SCP03 secure messaging. When SCP03 is selected AEAD is still necessary to protect the response of the authentication command message.

The Ciphers Suites defined by this specification are listed in Table 4-2. Each Cipher Suite is assigned a 2-byte identifier for reference in environments where UTF-8 strings cannot be used.

Table 4-2: Opacity Blinded Protocol Cipher Suites

Cipher Suites	2-byte Identifier
OPACITY_128_ECDHB_NIST_KDF2S_CCM_AEAD	'01 40'
OPACITY_128_ECDHB_NIST_KDF2S_CCM_SCP03	'01 60'
OPACITY_128_ECDHB_NIST_KDF2S_GCM_AEAD	'01 41'
OPACITY_128_ECDHB_NIST_KDF2S_GCM_SCP03	'01 61'
OPACITY_128_ECDHB_BR1_KDF2S_CCM_AEAD	'09 40'
OPACITY_128_ECDHB_BR1_KDF2S_CCM_SCP03	'09 60'
OPACITY_128_ECDHB_BR1_KDF2S_GCM_AEAD	'09 41'
OPACITY_128_ECDHB_BR1_KDF2S_GCM_SCP03	'09 61'
OPACITY_192_ECDHB_NIST_KDF2S_GCM_AEAD	'02 41'
OPACITY_192_ECDHB_NIST_KDF2S_GCM_SCP03	'02 61'
OPACITY_192_ECDHB_BR1_KDF2S_GCM_AEAD	'10 41'
OPACITY_192_ECDHB_BR1_KDF2S_GCM_SCP03	'10 61'
OPACITY_256_ECDHB_NIST_KDF2S_GCM_AEAD	'03 41'
OPACITY_256_ECDHB_NIST_KDF2S_GCM_SCP03	'03 61'
OPACITY_256_ECDHB_BR1_KDF2S_GCM_AEAD	'11 41'
OPACITY_256_ECDHB_BR1_KDF2S_GCM_SCP03	'11 61'
OPACITY_128_ECDHB_NIST_KDF2S_ETM_AE	'01 4A'
OPACITY_128_ECDHB_SM2_KDF2S_ETM_AE	'0A 4A'

Table 4-3: Opacity-ZKM Mode Cipher Suites

Cipher Suites	2-byte Identifier	P1 byte [ANSI 504-1] Table 91
OPACITY_128_ECDH_NIST_KDF1S_SCP03 (used in [NIST 800-73-4])	'01 04'	'27'
OPACITY_192_ECDH_NIST_KDF1S_SCP03 (used in [NIST 800-73-4])	'02 04'	'2B'

Table 4-4: Opacity-FS Mode Cipher Suites

Cipher Suites	2-byte Identifier	P1 byte [ANSI 504-1] Table 91
OPACITY_128_ECDH_NIST_KDF1S_SCP03	'01 04'	'28'
OPACITY_256_ECDH_NIST_KDF1S_SCP03	'02 04'	'2D'

4.2 Protocol Parameters

Table 4-5: Protocol Parameters

Name	Definition	Type	Format/Length
C_se	Certificate Chain authenticating the SE	X.509 or ISO 7816-8 CVC	
CB_h	Host control byte	See section 5.1	See Table 5-1.
CB_se	SE control byte	See section 5.1	See Table 5-1.
CS_se	Allowed Cipher Suite	See section 5.2	See Table 5-2.
CSL_h	Requested Cipher Suite List	See section 5.2	
d_se	SE private key	EC private key	Size of curve in bits
d_bse	Random Blinding factor. The size is set by the SE. It is between n_bitsec and N, the order of the EC curve. The host must support these values.	Byte string	The length is n1. The host must support a size that is as large as the order of the curve (size of private key).
dec_{k}(data, aad)	Decryption of data with Key k (reverse Authenticated encryption operation), aad is associated data that is not encrypted used in the AEAD MAC computation.	Byte string	
Default_CS	Default cipher suite requested from the terminal. At minimum the SE must use this cipher suite to generate the encryption key to encrypt the allowed cipher suite.	See section 5.2	See Table 5-2.
enc_{k}(data, aad)	Authenticated Encryption of data, and associated data aad with Key k	Byte string	
enc_se	Encrypted data generated on an SE	Byte string	Variable
G	EC Base Point	EC Point representation	See EC Point representation in section 3.1.4.
Head	Opacity Message Header	Byte String	Two bytes: Protocol Type and Version '0A01'
KR_h	Requested SE Key reference	See section 5.4	1 byte
KR_se	Allowed SE key reference	See section 5.4	1 byte
len_msg_se	Fixed length of SE messages	Integer	All SE messages should be padded to have exactly this length.
Msg_h	Full host message	Byte String	

Name	Definition	Type	Format/Length
N	EC private key length in bits	Integer	
n_bitsec	Security strength supported (in bits). Defined in [NIST 800-56A] section 5.5.1.2, Table 2.	Integer	128 for N=256 192 for N=384 256 for N=512 or 521
n1_h	Blinding factor length in bytes as requested from the host.	Integer See section 5.5	Must be higher or equal than n_bitsec. EC Curve key length is recommended for non-traceability.
n1_se	Random Blinding factor length in bytes chosen by the SE.	Integer	Must be higher or equal than n_bitsec. EC Curve key length is recommended for non-traceability.
n2_h	Host Session identifier length in bytes	Integer	8
n2_se	SE session identifier length	Integer	8
Pad_se	Padding for non-traceability. Padding should be used in the response message prior to encryption to remove any indication of mode or error state	Byte string	Length is determined to adjust the SE message length to a fixed size (len_msg_se). The leftmost bit of padding should be 1 followed with all 0s.
Payload	Application payload to protect – transmitted from the SE		
Q_bse	Blinded SE public key	EC Point representation	See EC Point representation in section 3.1.4
Q_eh = [d_eh] . G	Host ephemeral public key, matched with the corresponding ephemeral private key: d_eh	EC Point representation	See EC Point representation in section 3.1.4
Q_se = [d_se] . G	SE authentication public key, matched with the corresponding private key: d_se	EC Point representation	See EC Point representation in section 3.1.4
sID_h	Random session identifier	Byte string	Length = n2_h
sID_se	SE session identifier. Use n2_se leftmost bytes from the x-coordinate of Q_bse.	Byte string	Length = n2_se
sk_csQ	Cipher suite encryption key, encrypts CB_se || CS_se || KR_se	XOR key	4 bytes

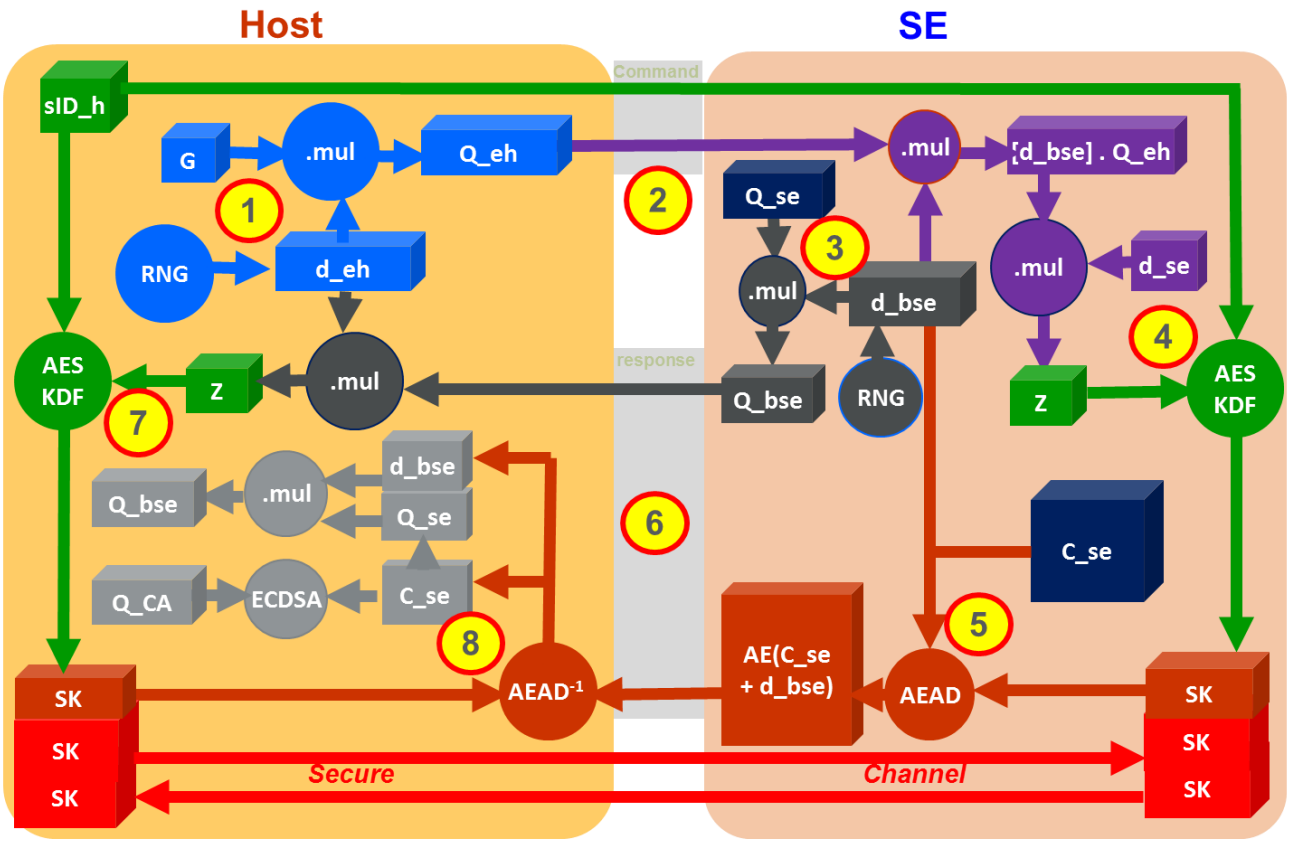
Name	Definition	Type	Format/Length
sk_h	Secure Messaging Session Keys for Authenticated encryption of information in transit from host to SE	AES key	Defined by cipher suite
sk_se	Secure Messaging Session Keys for Authenticated encryption of information in transit from SE to host	AES key	Defined by cipher suite
T_{len}(data)	Truncation of data: leftmost '*len*' bits.	Byte string	
x	Order in cipher suite list	integer	
Z	Intermediate Shared secrets (x-coordinate of ECDH shared resulting point)	EC point representation	
zero_key	AES key. All key bytes have a zero value (see [NIST 800-56C]).	AES key	

4.3 Protocol Flow

4.3.1 Principle

The following diagram provides an illustration of the protocol flow.

Figure 4-1: Protocol Flow Overview



1. Generate eph. key pair
2. Send one time id and eph pub key
3. Generate blinding factor. blind SE pub key
4. Compute ECDH+KDF – generate session keys
5. Prepare and encrypt response
6. Send blinded key and encrypted response
7. Compute ECDH +KDF with blinded key.
8. Decrypt and verify response. Authenticate Blinded key

Each side computes: Z = [d_bse . d_eh . d_se] .

Assumedly: d_se . G = Q_se

4.3.2 Protocol Steps

The Opacity Protocol consists of three required steps, followed by optional steps to exchange messages within the secure channel that is established.

1. **Host Request**: Host prepares and sends Command (i.e. Authentication Request) to SE
2. **SE Response**: SE receives and processes Command, prepares and sends Response to Host
3. **Host Validation**: Host receives and processes Response, authenticating SE and decrypting SE response payload
4. <Optional secure channel messages>

Table 4-6: Protocol Steps – Host Prerequisites

Step #	Description	Comment
H1	Known **CB_h** = BLIND Known **CSL_h** = **Default_CS**, followed with other Cipher Suites List Known **CF_h** = Certificate Format Known **KR_h** = Default Key reference, corresponding to **CSL_h**[0] Known **G**, **N** = Base point, EC key length Known **n1_h** = Desired blinding factor size Known **n2_h** = 8	Set host control parameter to Blinded protocol mode (see section 5.1). Set host Cipher Suite list to the values supported by the host. The first cipher suite element shall match the ephemeral public key sent by the host. The following cipher suite elements represent the alternate cipher suites that are allowed by the host, and that the SEs may support or require if the first cipher suite is not desirable for securing the transaction. Set default key reference to match default cipher suite (optional for non-proprietary use). This can only be specified for the default (first) cipher suite. The certificate Format byte indicates the different formats supported by the host. Set curve parameters G, N according to the first Cipher Suite element of CSL_h. KDF used for first key agreement must be Default CS KDF. Set desired blinding size (see section 5.5). Low values allow faster computation. 'FF' means a default size will be chosen by SE. SE may choose longer or full blinding size. Recommended host session identifier length is 8.

Table 4-7: Protocol Steps – SE Prerequisites

Step #	Description	Comment
SE1	Known n2_h = 8 Known all allowed values for (d_se , Q_se = [d_se] . G , Cert_se) Known Payload	n2: size of host identifier sID_h = 8 d_se, Q_se, Cert_se are set for all allowed curves. Obtain or prepare payload

Table 4-8: Protocol Steps – Host Request

Step #	Description	Comment
H2	Generate sID_h = RNG (n2)	Generate a host session identifier with the selected length (see section 3.1.2).
H3	Generate d_eh = RNG (N) Execute Q_eh = [d_eh] . G Validate Q_eh	Generate an ephemeral EC Key Pair for the Host. See section 3.1.2 for the recommended choice of a random number generator. Perform full public-key validation ([NIST 800-56A] section 5.6.2.3.2), either as part of the key generation process or as a separate process.
H4	Send msg_h = Head || **CB_h || sID_h || CF_h || CSL_h || KR_h || n1_h || Q_eh**	When SE is present in the field, send ephemeral public key, session ID, control byte, cipher suite list.
H5	Wait	Wait for SE response

Table 4-9: Protocol Steps – SE Response

Step #	Description	Comment
SE2	Receive msg_h = Head || **CB_h || sID_h || CF_h || CSL_h || KR_h || n1_h || Q_eh**	Obtained from host
SE3	if CB_h & BLIND == '00' return CB_ERROR if CB_h & CS_CHANGE then goto SE8 Check CF_h Set CB_se = CB_h | BLIND	Check if terminal request allows the SE to respond: • Check if Control byte CB_h is supported • If this is a retry, then assume that Q_eh complies with the CS_se returned in the previous response. Ignore **CSL_h || KR_h || n1_h** • Check if Cert_se has a format supported by the host in CF_h

Step #	Description	Comment
SE4	Set n1_se larger or equal to n1_h (shall be higher than n_bitsec) and N/8.	Set blinding factor size. n1_se according to n1_h. Special value '00' means no blinding and 'FF' means a default size shall be chosen by SE (see section 5.5). n1_se: size of blinding factor between 0 and N
SE5	Search x = order of allowed cipher suite in CSL_h if x does not exist, return CB_ERROR if SE does not support KDF and curve of CSL_h[0], return CB_ERROR otherwise Set CS_se = CSL_h[x]	Select allowed cipher suite from host cipher suite list. Determine by searching from left to right in the list if any cipher suite present in the host list is allowed. This also means that the SE has a corresponding valid key pair and certificate Cert_se The order starts from x=0 (Default_CS), the leftmost element in CSL_h. If no host cipher suite is allowed or if KR_h is incorrect, return CB_ERROR (recommended for additional security). Otherwise select allowed cipher suite.
SE6	if x != 0 and (CS_se Curve or CS_se KDF is incompatible with default CS) then CB_se |= CS_CHANGE, Payload = NULL, Cert_se = NULL	Check if the selected SE cipher suite selection requires the host to restart the protocol. If the requested default host cipher suite is not an allowed SE Cipher suite for the requested SE key reference (x != 0). It will however protect the response information with default_CS (CSL_h[0]). The SE indicates that a cipher suite change is requested. No certificate or application payload will be returned. The host will need to restart the protocol.
SE7	if x == 0 or x != 0 and curve is compatible with CS_se curve and KDF(Default_CS) is the same as KDF (CS_se) then continue	If the selected SE cipher suite does not require the host to restart the protocol, then select the appropriate SE certificate. SE can protect response with CS_se, Cert_se.

Step #	Description		Comment
SE8	Set KR_se Set G, N = Base point, EC key length. Set d_se , Q_se = [d_se] . G , Cert_se		Select SE key if x=0, Use KR_se = KR_h Determine SE private key, public key, and Certificate (d_se, Q_se, Cert_se) according to chosen cipher suite CS_se. Otherwise use KR_se based on Default_CS. Generate key pair (d_se, Q_se) if necessary.
SE9	Generate	d_bse = Rand (n1_se)	Generate blinding factor.
SE10	Execute	Q_bse = [d_bse] **.** Q_se	Create blinded public key.
SE11	Set	CB_se = BLIND	Set Opacity Mode selected by the SE.
SE12	Validate	Q_eh #belongs to EC domain	Check ephemeral host public key validity.
SE13	Execute	Z = [d_bse . d_se] **.** Q_eh	Compute shared secret.
SE14	Set	sID_se = T_{n2_se}(Q_bse)	Build SE Identifier for KDF.
SE15	Select	CS_se as cipher suite for further operations	KDF of Default_CS is selected to be the same as KDF of CS_se.
SE16	Execute	sk_cs || sk_se || sk_h = KDF (Z , info(sID_h, CB_h, Q_eh, sID_se), len)	Compute session keys. sk_cs is an XOR key with the same size as CB_se || CS_se || KR_se. KDF parameters are aligned with [ANSI 504-1] except for CB_se. **Note:** ISO secure messaging may be supported with the extension of KDF output with secure messaging keys (see section 4.5).
SE17	Compute	Pad_se	The padding should be computed so the whole result of the AEAD computation, including the plain text, cipher text, and MAC values, has a fixed size message length equal to len_msg_se. This padding value will avoid revealing the message modes and error states. The first bit of padding should be 1 followed with 0.
SE18	Execute	enc_cs = XOR (sk_cs, CB_se || CS_se || KR_se)	Compute encrypted cipher suite using XOR with sk_cs.

Step #	Description		Comment
SE19	Execute	enc_se = Enc_{sk_se} (n1_se || d_bse || Cert_se || Payload || Pad_se, msg_h || Q_bse || enc_cs)	Compute Authenticated Encryption cryptogram. The MAC is computed on the whole message sequence, starting from the terminal request message, i.e. Q_eh (message only, no transport or APDU header). **Note:** Cert_se and Payload may be NULL if the SE requests the host to restart the protocol (i.e. CB_se is set to CS_CHANGE).
SE20	Zeroize	Z, d_bse, sk_cs	Destroy current shared secret.
SE21	Send	Head || Q_bse || enc_cs || enc_se	Return to host.

Table 4-10: Protocol Steps – Host Validation

Step #	Description		Comment
H6	Receive Head || Q_bse || enc_cs || enc_se,		Q_bse is a point on the curve indicated by the SE cipher suite CS_se. Enc_cs is the encrypted cipher suite, key reference and control byte. enc_se is the authenticated encryption result including Cert_se, d_bse.
H7	set sID_se = T_{n2_se}(Q_bse)		Set SE identifier sID_se.
H8	Validate	Q_bse #belongs to EC domain	Check that the point is on the same curve as Q_eh.
H9	Execute	Z = [d_eh] . Q_bse	Compute the shared secret.
H10	Select	Default_CS as cipher suite for further operations	Used to determine the KDF.
H11	Execute	sk_cs || sk_se || sk_h = KDF (Z , info(sID_h, CB_h, Q_eh, sID_se), len)	Compute the session keys. KDF parameters are aligned with [ANSI 504-1] except for CB_se. The output session keys are sk_cs used to compute XOR on enc_cs, sk_se, used for AEAD and further communication from the card, and sk_h for further communication from the terminal. **Note:** ISO secure messaging may be supported with the extension of KDF output with secure messaging keys (see section 4.5).

Step #	Description		Comment
H12	Zeroize	Z, d_eh	Destroy Z, ephemeral private key
H13	Execute Zeroize	CB_se || CS_se || KR_se = XOR (sk_cs, enc_cs) sk_cs	Decrypt the cipher suite and key reference information.
H14	If CB_se & CS_CHANGE then Payload == NULL. restart protocol with correct KR_h, Default_CS Goto H1		If the SE requests a change of cipher suite that requires to restart the protocol. **Note:** There is no SE certificate to validate from the host in this case.
H15	Execute	n1_se || d_bse || Cert_se || Payload || Pad_se = Dec_{sk_se} (enc_se, msg_h || Q_bse || enc_cs)	Decrypt enc_se using AEAD. Use aad = msg_h || Q_bse || enc_cs to include integrity validation on both request and response messages.
H16	Extract Validate	Q_se = PubK (Cert_se) Q_se (belong to EC domain)	Recover SE public key.
H17	Validate	[d_bse] . Q_se = Q_bse	Confirm that the blinded public key that was received is actually the result of the blinding factor applied to the SE public key.
H18	Validate	Cert_se	To verify the SE certificate signature according to the cipher suite or directive within the certificate.

The Opacity Blinded protocol is completed and a secure channel is established. Use secure messaging with SK_h to protect additional commands and SK_se to decrypt and authenticate the responses (see section 6.2.3).

4.4 Protocol Parameter Negotiation

The Opacity protocol allows the negotiation between the host and the SE of the protocol mode, SE cipher suite, SE key reference, and SE Certificate Format.

4.4.1 Host Cipher Suite List

The host provides a mandatory cipher suite list in the command that informs the SE about the combinations of curve, secure messaging mode, KDF, and AEAD mode which can be accepted (see section 4.1). The first element of the host cipher suite list is the default cipher suite. The host ephemeral public key shall be on the default cipher suite curve.

4.4.2 SE Cipher Suite Selection

The SE must select one cipher suite of the host cipher suite list to successfully complete the protocol execution. If the SE application policy allows the use of the default cipher suite, then the SE should select it. However the appropriate selection of cipher suite and key reference is determined per SE application policy. If no cipher suite is supported or allowed from the host list, or if the SE does not support the default Curve and default KDF, then the SE will return an error with no further information.

The SE must support the default cipher suite curve and the default cipher suite KDF, to allow the protection of SE protocol parameters in the response. The SE Application policy may forbid to return identity data such as certificates or application payload data with the default curve. In that case the SE will indicate in a first response protected by the default curve, which other – non-default – cipher suite supported by the host shall be used to obtain the desired identity or application data.

4.4.3 Certificate Format

The host provides a certificate format byte (CF_h) in the command that informs the SE about all certificate formats that the host allows for authentication. The SE shall return a certificate having a format that the host indicates in the command. Otherwise it returns an error with no further information.

4.4.4 SE Key Reference

The host provides a SE key reference (KR_h) in the command that informs the SE about what private key should be used (d_se). KR_h shall refer to a key that belongs to the default cipher suite. The SE should use this key (i.e. KR_se = KR_h) if allowed by the SE application policy, if the SE has the private key corresponding to the proposed SE key reference (KR_h), and if the host default curve is the curve for KR_se.

4.4.5 SE Response Parameters

To inform the host about the selected cipher suite and key reference, and protect this information from eavesdropping, the SE must encrypt this information in the response and shall use the default cipher suite curve and KDF for that matter. Section 4.4.6 details the different protocol cases.

4.4.6 Protocol Cases

4.4.6.1 Case 1 – SE allows default cipher suite

In that case the SE may choose the private key (d_se) corresponding to the host key reference (KR_h) or a different private key for the same default cipher suite curve. The SE will indicate the chosen key reference in the response (KR_se). The SE uses all default cipher suite parameters, KDF, and AEAD mode, to produce the session key and protect the response information including the selected key agreement parameters, in that case the default cipher suite and selected key reference, the blinding factor, and the certificate.

4.4.6.2 Case 2 – SE supports default curve and KDF, but requires another suite in list

If the non-default cipher suite is compatible with the default suite (same curve and KDF), but with a different AEAD mode or secure messaging then the response may be fully protected with the non-default cipher suite.

If the non-default cipher suite has an incompatible curve or incompatible KDF with the default cipher suite, the protocol must be restarted with appropriate key agreement parameters selected by the SE.

The SE will generate an ephemeral key pair on the default curve, and use the default cipher suite to protect the response. The response will contain the SE ephemeral public key protected SE Key Agreement Parameters, but no certificate, blinding information, or SE application payload.

The host will decrypt the response to determine the selected cipher suite and key reference. It will check that the SE cipher suite selection is included in the cipher suite list and will resubmit a command with appropriate SE cipher suite, including a new ephemeral host public key. The host cipher suite list and default cipher suite must remain unchanged as the selected SE cipher suite must not be revealed.

4.4.6.3 Case 3 – SE supports default curve but does not allow any cipher suite in list

The SE is not able to protect its application payload using the host capabilities. It shall return an error.

4.4.6.4 Case 4 – SE does not support default curve and KDF

The SE is not able to protect its cipher suite selection in a response. It shall return an error.

4.5 Secure Messaging

The protocol allows a choice of secure messaging mechanisms (see section 5.2):

- AEAD (see section 3.4): The same AEAD scheme as indicated by the cipher suite is used to further protect the host commands and SE responses.
- SCP03 (per [GPCS-D]) or SCP03 ISO (per [GPCS-ISO]):
  - Three AES session keys are needed for Secure Messaging in addition to: SK_mac, SK_enc, and SK_rmac. The KDF (KDF1S or KDF2S) is adjusted to output the appropriate number of keys. Steps H11 in Table 4-10 and SE16 in Table 4-9 should be replaced as follows:

    sk_cs || sk_se || sk_h || sk_enc = KDF (Z, info(sID_h, CB_h, Q_eh, sID_se), len)

    where sk_se = sk_rmac and sk_h = sk_mac
  - The usage of SCP03 or SCP03 ISO variant would be defined in specific configuration documents or would be implicitly determined from the context (e.g. ISO SCP03 would be used if the Application is associated with an ISO SD as per [GPCS-ISO]).

5 Parameter Encoding

This chapter describes the encoding for the Opacity Blinded protocol parameters. The encoding of Opacity ZKM and Opacity FS parameters is specified in [ANSI 504-1].

5.1 Control Byte Encoding

Table 5-1: Control Byte Encoding

b8	b7	b6	b5	b4	b3	b2	b1	CB Encoding	Description
				(RFU)		(*)			
0	–	–	1	0	0	0	0	CS_CHANGE	Cipher suite must be changed
0	–	1	0	0	0	0	0	BLIND	Blind Mode requested
0	0	0	0	0	0	–	–	ZKM (*)	ZKM mode requested
0	1	0	0	0	0	–	–	FS (*)	Full Secrecy Mode requested
1	0	0	0	0	0	0	0	ERROR	Error occurred

(*) As defined in [ANSI 504-1].

5.2 Cipher Suite and Cipher Suite List Encoding

When referenced in an opacity protocol response message, a unique Cipher Suite descriptor is encoded on two bytes as follows:

CS_se = '00' || CurveByte || CipherByte (see CurveByte in Table 5-2 and CipherByte in Table 5-3)

When referenced in an opacity protocol command message, a Cipher Suite List descriptor is encoded on two or more bytes as follows:

CSL_h = '00' || CurveByte_0 || CipherByte_0 || … || CurveByte_n || CipherByte_n || 'FF'

CSL_h describes the curves and cipher suites that are supported by the host or terminal. The leftmost cipher suite (CurveByte_0 || CipherByte_0) is the default cipher suite that the SE must support to provide a first protected response.

For instance, if the host supports NIST P-256 with either AEAD GCM and AEAD CCM and 1-step KDF and AEAD Secure Messaging, or NIST P-384 with AEAD CCM only. The host also supports both X.509 and CVC certificate format.

CSL_h = '00' || {NIST_P-256} || {AEAD-GCM} || {NIST_P-256} || {AEAD-CCM} || {NIST_P384} || {AEAD-CCM} || 'FF'

Where according to Table 5-2, NIST_P-256 = '01', NIST_P-384 = '02', and according to Table 5-3, AEAD-CCM = '00', AEAD-GCM = '01'

CSL_h = '00010101000200FF'

CS_se response shall only contain a single curve and a single cipher suite that are selected among the combinations that are proposed in CSL_h.

For instance:

CS_se = '000100'

Table 5-2: Cipher Suite Encoding – Curve Byte

Description	Value
Reserved	'00', 'FF'
Proprietary	'80' – 'FE' (range)
NIST P256	'01'
NIST P384	'02'
NIST P521	'03'
Brainpool 256r1	'04'
Brainpool 384r1	'05'
Brainpool 512r1	'06'
Brainpool 256t1	'07'
Brainpool 384t1	'08'
Brainpool 512t1	'09'
SM2 Fp-256	'0A'
RFU	'0B' – '7F' (range)

Table 5-3: Cipher Suite Encoding – Cipher Byte

b8	b7	b6	b5	b4	b3	b2	b1	Cipher Suite Cipher Byte Encoding
–	–	–	–	–	X	X	X	AEAD mode (unspecified values are RFU)
–	–	–	–	–	0	0	0	AEAD-CCM
–	–	–	–	–	0	0	1	AEAD-GCM
–	–	–	–	–	0	1	0	AEAD-ETM-CTR-CMAC See Table 3-5
–	–	–	–	–	0	1	1	AEAD-EAX
–	–	–	–	–	1	0	0	AEAD-OCB2
–	–	X	X	X	–	–	–	secure messaging
–	–	0	0	0	–	–	–	AEAD secure messaging
–	–	0	0	1	–	–	–	AE secure messaging
–	–	1	0	0	–	–	–	SCP03 secure messaging
–	–	1	0	1	–	–	–	SCP03 ISO secure messaging
X	X	–	–	–	–	–	–	KDF
0	0	–	–	–	–	–	–	1-step KDF
0	1	–	–	–	–	–	–	2-step KDF

5.3 Certificate Format

Table 5-4: Cert Format Byte

b8 (RFU)	b7 (RFU)	b6 (RFU)	b5 (RFU)	b4 (RFU)	b3 (RFU)	b2	b1	Cert Format Byte Encoding
				X	X	X	X	Cert Format Encoding
–	–	–	–	0	0	0	1	CVC encoding
–	–	–	–	0	0	1	0	X.509 encoding
–	–	X	X	–	–	–	–	EC Point Representation
–	–	0	0	–	–	–	–	As defined in [TR 03111] section 3.2
–	–	0	1	–	–	–	–	Optimized. See section 3.1.4

5.4 Key Reference

Table 5-5: Key Reference

Description	Value
Reserved	'00', '80' – 'FF'
Application Specific	'01' – '7F'

The key reference refers to the SE ECC static private key.

5.5 Blinding Factor Policy

A random blinding factor is used in Opacity to obfuscate the SE public key. The size of the blinding factor determines the strength of obfuscation (i.e. level of privacy). The recommended size is the EC curve key length.

The blinding factor policy determines the default and minimum values for the blinding factor.

The blinding factor policy n1_h is encoded on one byte as follows:

- 'FF': The blinding factor size will be a default size chosen by the SE.
- 'n_bitsec/8 to N/8' (N: EC curve order): The actual size in bytes of the blinding factor requested by the host.

6 Messaging

6.1 Message Flow

The message flow for all Opacity protocols (ZKM, FS, and Blinded) consists of a first command message transmitted by a sender entity (host) to a responder entity (Secure Element). This is followed by a response message transmitted from the responder to the sender. Secure channel key establishment is completed at this point, so any number of protected messages may be transmitted both ways (see Figure 2-1).

6.2 ISO 7816-8 Based Messaging

The data format, message format, and secure messaging for Opacity ZKM and Opacity FS protocols are specified in [ANSI 504-1].

This section defines the data format, message format, and secure messaging for the Opacity Blinded protocol.

6.2.1 Data Format

6.2.1.1 Command Message Data

Table 6-1: Command Data

Name	Definition	Length
CB_h	Host control byte (see section 5.1)	1 byte
sID_h	Host identifier	8 bytes
CF_h	Certificate Format Byte (see section 5.3)	1 byte
CSL_h	Host cipher suite list (see section 4.4.1). 'FF' if none.	(2*n+2) bytes. n: number of cipher suites
KR_h	SE key reference for default cipher suite (see section 5.4)	1 byte
n1_h	Blinding factor size/policy (see section 5.5)	1 byte
Q_eh	Ephemeral host public key encoded as described by the certificate format byte in section 5.3	Var.

6.2.1.2 Response Message Data

Table 6-2: Response Data

Name	Definition	Length
Q_bse	Blinded public key (same EC point format and length as Q_eh) encoded as described by the certificate format byte in section 5.3	Var.
enc_cs	Cipher text produced by XORing CB_se || CS_se || KR_se	4 bytes
enc_se	Cipher text produced via authenticated encryption. Includes mac cryptogram computed over both command and response.	Var.

6.2.2 GENERAL AUTHENTICATE APDU

This section applies to the Opacity Blinded protocol only. Transport for Opacity ZKM and FS protocols is specified in [ANSI 504-1].

6.2.2.1 Definition and Scope

The GENERAL AUTHENTICATE command defined in [ISO 7816-4] is used to transport the Opacity command message and the Opacity response message between the card and the host.

This command also initiates an optional Secure Channel Session.

At any time during a current Secure Channel, the GENERAL AUTHENTICATE command can be issued to the card in order to initiate a new Secure Channel Session.

6.2.2.2 Command Message

The GENERAL AUTHENTICATE command message is coded according to the following table:

Table 6-3: GENERAL AUTHENTICATE Command Message

Code	Value	Meaning
CLA	'00'	See [GPCS] section 11.1.4.
INS	'86'	GENERAL AUTHENTICATE
P1	'xx'	Must be equal to the one byte cipher suite value for Opacity ZKM or Opacity FS. See [ANSI 504-1] P1 byte in tables 4-3 and 4-4. Must be equal to 'FF' when Authenticated Encryption without associated data is used for secure messaging, the APDU header is not protected by the secure messaging. Must be equal to '00' for all other cases.
P2	'xx'	Opacity SE Private Key d_se reference or '00'. Must be equal to '00' (implicit) or KR_h.
Lc	'xx'	Length of command data in bytes.
Data	'xx xx .. '	Command data (See section 6.2.1.1.)
Le	'00'	

6.2.2.3 Reference Control Parameter P1

This value should be set to '00'. If not '00' it must match the default recommended host cipher suite, i.e. the leftmost byte of CSL_h in command data.

6.2.2.4 Reference Control Parameter P2

The Key Reference should be '00'. If non zero, it must be set to KR_h.

6.2.2.5 Data Field Sent in the Command Message

See section 6.1.

6.2.2.6 Response Message

The data field of the response message shall be as specified in section 6.2.1.2.

6.2.2.7 Processing State Returned in the Response Message

A successful execution of the command shall be indicated by status bytes '90' '00'. This command may either return a general error condition as listed in [GPCS] section 11.1.3 or the following error conditions:

Table 6-4: GENERAL AUTHENTICATE Error Conditions

SW1	SW2	Meaning
'6A'	'88'	Referenced data not found

6.2.3 Secure Messaging

The secure messaging for Opacity ZKM and Opacity FS protocols is specified in [ANSI 504-1].

The following sections define secure messaging mechanisms and format for the Opacity Blinded protocol.

6.2.3.1 AEAD

The AEAD scheme referenced in the cipher suite is used to further protect the host commands and SE responses.

The APDU protection may leverage AEAD using AE modes defined in section 3.4.3 as follows:

AD = APDU Header = (CLA || INS || P1 || P2)

PT_c = APDU command Payload (including TLV)

CT_c = Encrypted command payload

C-APDU = AD || PT_c

MAC_c = command MAC (defined from AEAD mode)

PT_r = APDU response Payload (including TLV)

CT_r = Encrypted response payload

MAC_r = response MAC (defined from AEAD mode)

SW = SW1 || SW2

R-APDU = PT_r || SW

Example Protection of C-APDU using SK_h:

AEAD (SK_h, AD, PT_c) = AD || CT_c || MAC_c = APDU Header || Encrypted payload || MAC_c

Example Protection of R-APDU using SK_se:

AEAD (SK_se, SW+MAC_c, PT_r) = SW || MAC_c || CT_r || MAC_r

The response is then reformatted as CT_r || MAC_r || SW.

Note: MAC_c is not transmitted but necessary to verify the MAC value.

Note: Certain AEAD modes (AEAD-CCM) require the Initialization Vector (IV) to be updated each time the same session key is reused (for instance a counter must be used as IV).

6.2.3.2 AE

The Authenticated Encryption (AE) Secure messaging scheme is identical to AEAD Secure Messaging (see section 6.2.3.1), with the following condition:

AD = NULL

SW = NULL

6.2.3.3 SCP03

SCP03 secure messaging mechanism and format is described in [GPCS-D]. Also see section 4.5 for SCP03 key establishment with Opacity.

If the SM2 curve is used, SMS4 block cipher shall be used instead of AES in the SCP03 protocol.

6.2.3.4 SCP03 ISO

SCP03 ISO secure messaging mechanism and format is described in [GPCS-ISO]. Also see section 4.5 for SCP03 ISO key establishment with Opacity.

If the SM2 curve is used, SMS4 block cipher shall be used instead of AES in the SCP03 ISO protocol.

6.3 Other Messaging Formats

6.3.1 Non APDUs

Non APDU schemes may be defined to transport Opacity messages if they preserve the overall framework and message flow but are outside the scope of this specification.

6.3.2 Non ISO APDUs

APDU commands that are not specified within [ISO 7816-4] may be defined to transport Opacity messages if they preserve the overall framework and message flow but are outside the scope of this specification.

7 Indication of Card Capabilities

7.1 Card Capability Information

Card Capability Information is defined in [GPCS] section 7.4.1.4 and section H.4.

Table 7-1 defines additional conditional contents for the ‘SCP Information’ TLV described in [GPCS] Table H-4.

Table 7-1: SCP Information

Tag	Length	Data / Description	Presence
'A0'	Variable	SCP information	
…	…	…	…
'85'	Variable	Supported cipher suites for SCP22	Conditional

The new ‘Supported cipher suites for SCP22’ TLV shall be used to describe the SCP22 cipher suites supported by the implementation and contains a sequence of supported cipher suite numbers as defined in Table 4-2.

8 Security Domain Support

8.1 Implementation Options

The Opacity protocol option byte “i” is defined from Table 5-1 as follows:

- “i” = 0: Opacity ZKM
- “i” = 20: Opacity Blinded
- “i” = 40: Opacity FS

8.2 Key and Certificate Requirements

This section indicates the list of keys and certificates that must be provided and personalized with a security domain supporting SCP22.

Table 8-1: Key and Certificate Requirements

Name	Reference	Comment
SE Authentication Key pair	KR_se (for each key pair d_se, Q_se)	One or multiple key pairs per cipher suite supported, referenced with KR_se
SE_certificate or certificate chain	Cert_se	One per public key and key reference KR_se