# Secure Messaging for OpenPGP card version 3.x

October 21, 2016

## Introduction

OpenPGP card specification [8] defines a mean to ensure integrity and confidentiality of communications with the OpenPGP card application. This concept, called Secure Messaging, is well known and widely used in the smart card ecosystem[1, 5]. However, the Secure Messaging defined in OpenPGP card specification is not as secure as other Secure Messaging protocols such as *Secure Channel Protocols (SCP)* from Global Platform specifications, and is not intented to protect communications only.

The most severe issue with Secure Messaging of OpenPGP card is the mutual authentication between an *off card entity (OCE)* (*i.e.* computer, smartphone, ...) and a OpenPGP card based on static AES keys. If these keys get compromised, forward secrecy of communications is not ensured, and man-in-the-middle attacks are easily practicable. Moreover, this scheme is not convenient to use in practice. For a card with no keys set, the OCE can generates such keys and store them in the card by transmitting them in clear after correct verification of PW3 (admin PIN), also transmitted in clear. For a card with some keys set, the OCE must also know them *a priori* in order to use them for Secure Messaging. However, it means that these keys exist on each OCE, and so for each card, assuming that each card has its own keys. Both cases are neither practicable nor secure.

In addition to the problem of using static keys, the Secure Messaging feature described in OpenPGP card specification imposes to use a new *initialization vector (IV)* generated by the card between each command/response. Concretely, before each command that needs to be protected the OCE must ask the card for a fresh IV (*a fortiori* transmitted in clear), which adds a strong overhead in terms of communications with the card.

Finally, the OpenPGP card specification says that a command protected with Secure Messaging as the same level of privileges that a non-protected command executed after successful presentation of PW3 (admin PIN), including the ability to reset the try counter of PW1 (user PIN), to override existing PGP keys and certificates, but also to set static keys for Secure Messaging. Since Secure Messaging grants admin privileges, it cannot be used to protect integrity and confidentiality of commands/responses only.

For these reasons, Secure Messaging feature of OpenPGP specification is more an alternative way of administrating a card to the traditional verification of PW3 (admin PIN) than a mean to protect communications with an OpenPGP card. In this document, we propose to replace the existing Secure Messaging feature of OpenPGP card specification by a state-of-the-art Secure Messaging feature to protect communications with a card only. Basically, we propose to reuse an existing *Secure Channel Protocol (SCP)* from Global Platform specifications with as few modifications as possible to be used with the current OpenPGP card specification.

## 1 Secure Channel Protocol

Global platform specifies several secure channel protocols with variable security guarantees. Most of these protocols provide a mutual authentication between a card and an OCE based on a shared secret such as a set of static DES (SCP01 and SCP02) or AES (SCP03) keys. Relying on a shared secret is not

convenient in practice as it requires to spread this secret among all OCEs, and so for each card. The only secure channel procotol specified by Global Platform not relying on a shared secret is the SCP11.

### 1.1 SCP11 overview

Two variants of SCP11 exist: SCP11a which allows mutual authentication of an OCE and a card, and SCP11b which handles only card authentication by an OCE.

The authentication of an OCE by a card in SCP11a requires that each OCE as its own certificate that each card can accept directly and/or can verify it according to a known Certification Authority. This scheme is not convenient in the context of a OpenPGP card as it requires to deploy a certificate and its corresponding private key on each OCE. Then, it is not better than SCP11b in terms of security in our context. Actually, as SCP11a contains no revocation mechanism, it is equivalent to not having OCE authentication as soon as an "entity" is compromised.

For these reasons, SCP11b is a good compromise for OpenPGP. The static private key embedded in an card is as secure as other OpenPGP keys it already handles, and it is possible to generate this key on board. The security of the protocol thus relies only on protection of the private key of the certification authority that is signing the card's certificate, and in the integrity of this certificate on OCEs.

### 1.2 Details of SCP11b

The specification of SCP11b is splitted in several documents [2, 3, 4] as it involves mechanisms from other secure channel protocols defined by Global Platform. The key agreement protocol used in SCP11b [4], which is specific to SCP11, is a variant of the ElGamal key agreement protocol. On the contrary, the Secure Messaging part of SCP11 is exactly the one used in SCP03 [2]. It involves AES in CBC mode for command/response encryption, and CMAC [7] with AES for command/response integrity.

Figure 1 depicts the key agreement part of SCP11b with notations detailed in table 1 (section 5.1 in [4]). All keypairs denoted in 1 use the same elliptic curve. In practice, the this curve is implicit known by both entities as it is the elliptic curve used for the static keypair ($K^s_{card}$, $K^p_{card}$) of the card. This curve also detemines the size of AES keys (section 5.2 in [4]) for the Secure Messaging part : if size of an EC point on the curve is strictly lower than $< 512$ bits, then generated session AES keys must be at most 128 bits long, otherwise these keys are 256 bits long.

## 2 Implementing SCP11b in OpenPGP card

The commands from SCP11b specification cannot be integrated as is in OpenPGP specification: some commands required for SCP11b already exist with a different meaning in OpenPGP card specification, some parameters of these commands have no meaning in our context, or some parameters have been specified differently in SCP11b and in OpenPGP card specification.

To implement SCP11b, the following actions are required:

- to store an authentication certificate and the corresponding private key in a card;
- to request a card for its authentication certificate and the corresponding public key;
- to initialize a Secure Messaging session, *i.e.* authenticate a card and establish session keys as depicted in figure 1.

Each of these actions can be implemented by an APDU command (section 6 in [4]), as described in sections 2.1 and 2.2 of this document.

The Secure Messaging part of SCP11b, that is encryption/decryption and MAC computation/verification of commands and responses, does not require specific commands. The signalisation bit(s) in

| | |
|---|---|
| $K^{s}_{card}$ | Static private key of the card |
| $K^{p}_{card}$ | Static public key of the card |
| $Cert_{card}$ | Certificate corresponding to the keypair ($K^{s}_{card}$, $K^{p}_{card}$) |
| $eK^{s}_{OCE}$ | Ephemeral private key of the OCE |
| $eK^{p}_{OCE}$ | Ephemeral private key of the OCE |
| $eK^{s}_{card}$ | Ephemeral private key of the card |
| $eK^{p}_{card}$ | Ephemeral public key of the card |
| ShSs | Shared secret generated involving the static key of the card |
| ShSe | Shared secret generated involving only ephemeral keys |

Table 1: Notations for keys and secrets involved in SCP11b.

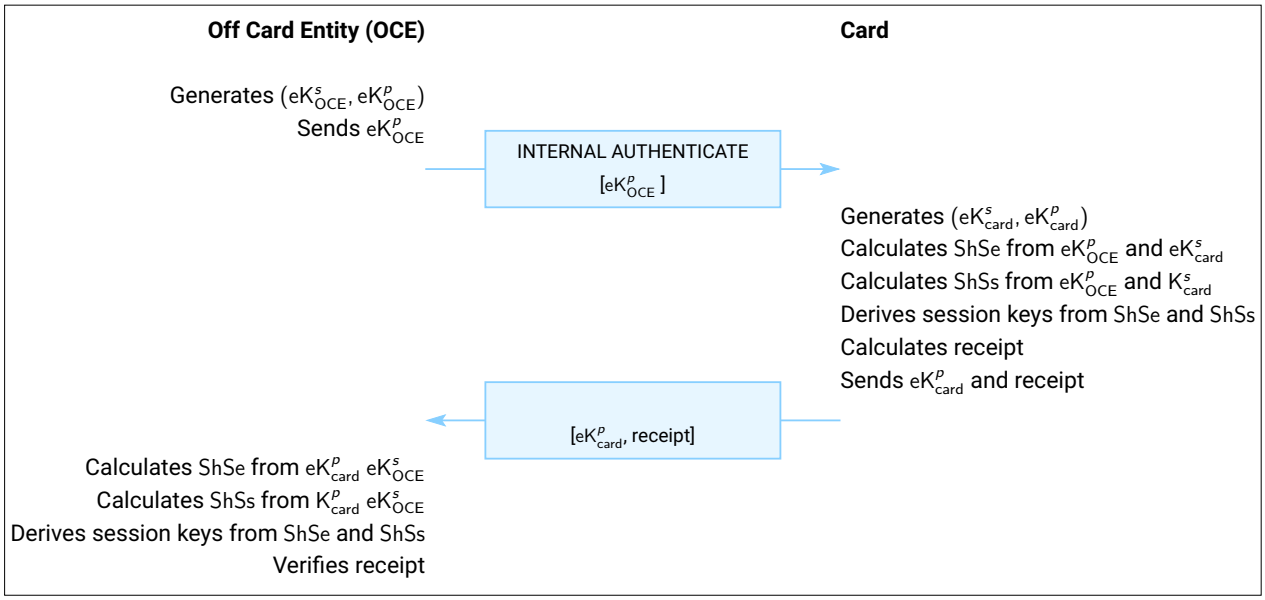


Figure 1: SCP11b key agreement.

the CLA byte of commands protected by Secure Messaging differs between Global Platform and OpenPGP specifications. In order not to interfere with existing implementations using the Secure Messaging feature of OpenPGP cards, the CLA byte is altered as described in section 2.3.

## 2.1 Card provisioning

While the certificate $\mathsf{Cert}_{\mathsf{card}}$ must be generated and signed out of the card, the corresponding keypair ($\mathsf{K}^{s}_{\mathsf{card}}$, $\mathsf{K}^{p}_{\mathsf{card}}$) can either be generated out of the card and then imported in the card, or generated by the card itself. The generation of the certificate and its signature by a trusted entity is out of the scope of this document. However, it is important to note that if the keypair is generated out of the card, it must be an EC keypair for a curve supported by OpenPGP specification.

The table 2 gives the new descriptors introduced in OpenPGP card specification to denote the new objects required for SCP11b implementation. These descriptors permit to handle the corresponding objects as other objects of the same kind (*i.e.* keypairs, certificates) in OpenPGP card specification (sections 4.4.1 and 4.4.2 in [8]). Each key is assigned a *control reference template (CRT)* (section 4.4.3.9 in [8]) and some algorithm attributes (section 4.4.3.7 in [8]). The CRT `A600` (section 6.2 in [4]) is used to reference the card authentication static keypair ($\mathsf{K}^{s}_{\mathsf{card}}$, $\mathsf{K}^{p}_{\mathsf{card}}$), and the tag `D4` for the corresponding algorithm attributes. The card authentication certificate $\mathsf{Cert}_{\mathsf{card}}$ corresponding to this keypair is referenced by the tag `7F21` as the fourth occurrence.

| | |
|---|---|
| `A600` | Control reference template (CRT) for the card static keypair ($\mathsf{K}^{s}_{\mathsf{card}}$, $\mathsf{K}^{p}_{\mathsf{card}}$) |
| `D4` | Tag for the algorithm attributes of the card static keypair ($\mathsf{K}^{s}_{\mathsf{card}}$, $\mathsf{K}^{p}_{\mathsf{card}}$) |

Table 2: New descriptors for SPC11b implementation in OpenPGP card specification.

### 2.1.1 Setting algorithm attributes for ($\mathsf{K}^{s}_{\mathsf{card}}$, $\mathsf{K}^{p}_{\mathsf{card}}$)

Before a keypair is generated or imported, the corresponding algorithm attributes must be set. The algorithm attributes for the keypair ($\mathsf{K}^{s}_{\mathsf{card}}$, $\mathsf{K}^{p}_{\mathsf{card}}$) are set via the PUT DATA command with INS=`DA` and P1P2=`D4` (section 7.2.8 in [8]). The algorithm attributes for this keypair must be valid ECDH attributes (section 4.4.3.7 in [8]), otherwise the card must issue an error and must not modify existing values.

### 2.1.2 Generating/importing ($\mathsf{K}^{s}_{\mathsf{card}}$, $\mathsf{K}^{p}_{\mathsf{card}}$)

The card authentication keypair can be generated via the GENERATE ASYMMETRIC KEY PAIR command with P1=`80` and the data field set to `A600` (section 7.2.13 in [8]). If the command exits normally, the public key of the generated keypair is returned (see section 2.1.5).

The card authentication keypair can be imported via the PUT DATA command with INS=`DB`, P1=`3F`, P2=`FF` and the data field set to the extended header list for ECDH with CRT set to `A600` (sections 4.4.3.9 and 7.2.8 in [8]).

### 2.1.3 Importing $\mathsf{Cert}_{\mathsf{card}}$ into the card

The card authentication certificate $\mathsf{Cert}_{\mathsf{card}}$ can be imported via a sequence of two commands:

1. SELECT DATA with P1=`03`, P2=`04` and the data field set to `60045C027F21` (section 7.2.5 in [8]);
2. PUT DATA with INS=`DA` and P1P2=`7F21` (section 7.2.8 in [8]).

The first command selects the fourth occurrence for the tag `7F21`, which is used to identify certificates. The second command actually imports the certificate.

### 2.1.4 Reading $\mathsf{Cert}_{\mathsf{card}}$ from the card

The card authentication certificate can be retrieved at any time via a sequence of two commands:

1. SELECT DATA with P1=`03`, P2=`04` and the data field set to `60045C027F21` (section 7.2.5 in [8]);
2. GET DATA with P1P2=`7F21` (section 7.2.6 in [8]).

The first command selects the fourth occurrence for the tag `7F21`, which is used to identify certificates. The second command actually reads the certificate.

### 2.1.5 Reading only $\mathsf{K}^{\mathit{p}}_{\mathsf{card}}$ from the card

The card authentication public key $\mathsf{K}^{\mathit{p}}_{\mathsf{card}}$ can be retrieved at any time via the GENERATE ASYMMETRIC KEY PAIR command with P1=`81` and the data field set to `A600` (section 7.2.13 in [8]).

## 2.2 Authentication and key agreement

The INTERNAL AUTHENTICATE is already specified in the OpenPGP card specification (section 7.2.12 in [8]), but for another purpose, when P1 = P2 = 0. Since this is the only behavior described in the specification, other values of P1 and P2 can be used to implement the key agreement part of SCP11b depicted in section 1.

On the contrary to the existing INTERNAL AUTHENTICATE command with P1P2=`0000`, this INTERNAL AUTHENTICATE command does not require the verification of PW2 (user PIN for signature).

The authentication and key agreement is achieved by issuing the INTERNAL AUTHENTICATE command with P1P2=`0100` and the data field set to the following content (section 6.5.2.3 in [4]):

<table>
<tr><th>Tag</th><th>Length</th><th colspan="3">Value description</th></tr>
<tr><td rowspan="6">A6</td><td rowspan="6">13</td><td colspan="3">Control reference template</td></tr>
<tr><th>Tag</th><th>Length</th><th>Value description</th></tr>
<tr><td>90</td><td>2</td><td>1100 (SCP11b without IDs)</td></tr>
<tr><td>95</td><td>1</td><td>3C (MAC + encryption)</td></tr>
<tr><td>80</td><td>1</td><td>88 (AES)</td></tr>
<tr><td>81</td><td>1</td><td>10 or 20 (AES key length L in bytes)</td></tr>
<tr><td>5F49</td><td>var.</td><td colspan="3">$\mathsf{eK}^{\mathit{p}}_{\mathsf{OCE}}$</td></tr>
</table>

The card must respond (section 6.5.3 in [4]) with:

| Tag | Length | Value description |
|---|---|---|
| 5F49 | var. | $\mathsf{eK}^{\mathit{p}}_{\mathsf{card}}$ |
| 86 | 16 | Receipt |

The two following sections describe how the session keys are derivated and the receipt is computed.

### 2.2.1 Derivation of session keys

From section 3.1.2 of [4], the key derivation to be used is the *X9.63 key derivation function* described in section 4.3.3 of [6] with the following parameters set (section 6.5.2.3 in [4]):

- SHA-256 is systematically used for $H$;
- the shared secret $\mathsf{ShS} = \mathsf{ShSe} \,||\, \mathsf{ShSs}$;
- the counter is initialized with `00000001`;
- the SharedInfo is the concatenation of

- `3C` : key usage qualifier is MAC + encryption;
- `88` : key ype is AES;
- `10` or `20` : key length ($L$) is 16 or 32 bytes long for AES 128 or 256 bits, respectively.

As depicted in section 5.2 of [4], the size of the keys to be generated depends on the size of a point on the elliptic curve used.

Four session keys have to be generated: the receipt key, S-ENC, S-MAC and S-RMAC. The value of the parameter $\kappa$ (section 4.3.3 in [6]) is thus $4 \times 128 = 512$ for AES 128 bits session keys, or $4 \times 256 = 1024$ for AES 256 bits session keys. The value of KeyData obtained from the derivation function is the concatenation of the generated session keys such that:

| **KeyData** | **Key** |
|---|---|
| 1 to $L$ | Receipt key |
| $L+1$ to $2L$ | S-ENC for command and response encryption |
| $2L+1$ to $3L$ | S-MAC for command MAC computation |
| $3L+1$ to $4L$ | S-RMAC for response MAC computation |

### 2.2.2 Receipt computation

The receipt consists in the CMAC [7] of the following input computed with the receipt key (section 5 in [5]):

| **Tag** | **Length** | **Value description** | | |
|---|---|---|---|---|
| `A6` | 13 | Control reference template | | |
| | | **Tag** | **Length** | **Value description** |
| | | `90` | 2 | `1100` (SCP11b without IDs) |
| | | `95` | 1 | `3C` (MAC + encryption) |
| | | `80` | 1 | `88` (AES) |
| | | `81` | 1 | `10` or `20` (AES key length $L$ in bytes) |
| `5F49` | var. | $\mathrm{eK}^{p}_{\mathrm{OCE}}$ | | |
| `5F49` | var. | $\mathrm{eK}^{p}_{\mathrm{card}}$ | | |

## 2.3 Protecting commands and responses

Protection of commands and responses is applied as described in section 5.3.2 in [4] (involving sections 6.2.6 and 6.2.7 of [2]) for the single case where both commands and responses are protected in integrity and confidentiality. However, since the OpenPGP card specification already uses some bits in the CLA byte to mark commands protected with Secure Messaging, and since we do not want to interfere with existing implementations, we propose to set the most significant bit in the CLA byte to mark commands protected with our Secure Messaging feature. So the CLA byte equals to `04` for commands without chaining, and to `14` for commands with chaining.

In case of commands chaining, the encryption and the MAC must be computed on the complete command before it is send splitted in several commands without additional protection for each individual command. The same scheme applies for chained responses.

It is important to respect the following rules to benefit the security brought by SCP11b. If a command is protected by Secure Messaging, then the response of the card must also be protected. If a complete command is received by the card without Secure Messaging protection while a Secure Messaging session is established (*i.e.* valid session keys exist), then the existing session keys must be forgotten so that new session keys will have to be negociated. In case of commands chaining, only the CLA byte of the last command should be used to determine if the content is protected by Secure Messaging or not.

## References


[1] ISO 7816-4 – Secure Messaging. http://www.cardwerk.com/smartcards/smartcard_standard_ISO7816-4_5_basic_organizations.aspx#chap5_6.

[2] GlobalPlatform Card, Secure Channel Protocol 03, Card Specification v2.2 – Amendment D, version 1.1.1 (document reference GPC_SPE_014). July 2014.

[3] GlobalPlatform Card, Security Upgrade for Card Content Management, Card Specification v2.2 – Amendment E, version 1.0.1 (document reference GPC_SPE_042). July 2014.

[4] GlobalPlatform Card, Secure Channel Protocol 11, Card Specification v2.2 – Amendment F, version 1.0 (document reference GPC_SPE_093). May 2015.

[5] GlobalPlatform, Card Specification, version 2.3 (document reference GPC_SPE_034). October 2015.

[6] BSI. Elliptic Curve Cryptography, Technical Guideline TR-03111, version 2.0. June 2012.

[7] NIST. Recommendation for Block Ciper Modes of Operation: The CMAC Mode for Authentication. May 2005.

[8] A. Pietig. OpenPGP Card specification – version 3.2.