CORE Working Group
Internet Draft
Intended status: Experimental

P. Urien
Telecom ParisTech

January 31 2014

Expires: August 2014

Remote APDU Call Secure (RACS)
draft-urien-core-racs-01.txt

Abstract

This document describes the Remote APDU Call Protocol Secure (RACS) protocol, dedicated to Grid of Secure Elements (GoSE). These servers host Secure Elements (SE), i.e. tamper resistant chips offering secure storage and cryptographic resources.

Secure Elements are microcontrollers whose chip area is about 25mm2; they deliver trusted computing services in constrained environments.

RACS supports commands for GoSE inventory and data exchange with secure elements. It is designed according to the representational State Transfer (REST) architecture. RACS resources are identified by dedicated URIs. An HTTP interface is also supported.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

Status of this Memo

.

Table of Contents

1 Overview

This document describes the Remote APDU Call Protocol Secure (RACS) protocol, dedicated to Grids of Secure Elements (GoSE). These servers host Secure Elements (SE), i.e. tamper resistant chips offering secure storage and cryptographic resources.

Secure Elements are microcontrollers whose chip area is about 25mm2; they deliver trusted computing services in constrained environments.

RACS supports commands for GoSE inventory and data exchange with secure elements.

RACS is designed according to the representational State Transfer (REST) architecture [REST], which encompasses the following features:
- Client-Server architecture.
- Stateless interaction.
- Cache operation on the client side.
- Uniform interface.
- Layered system.
- Code On Demand.

1.1 What is a Secure Element

A Secure Element (SE) is a tamper resistant microcontroller equipped with host interfaces such as [ISO7816], SPI (Serial Peripheral Interface) or I2C (Inter Integrated Circuit).

The typical area size of these electronic chips is about 25mm2. They comprise CPU (8, 16, 32 bits), ROM (a few hundred KB), nonvolatile memory (EEPROM, FLASH, a few hundred KB) and RAM (a few ten KB). Security is enforced by multiple hardware and logical countermeasures.

According to the [EUROSMART] association height billion of such secure devices were shipped in 2013. Secure elements are widely deployed for electronic payment (EMV cards), telecommunication (SIM modules), identity (electronic passports), ticketing, and access control.

Most of secure elements include a Java Virtual Machine and therefore are able to execute embedded program written in the JAVACARD language. Because these devices are dedicated to security purposes they support numerous cryptographic resources such as digest functions (MD5, SHA1, SHA2...), symmetric cipher (3xDES, AES) or asymmetric procedures (RSA, ECC).

A set of Global Platform [GP] standards control the lifecycle of embedded software, i.e. application downloading, activation and deletion.

As an illustration a typical Secure Element has the following characteristics:

- JAVACARD operating system;
- Compliant with the GP (Global Platform) standards;
- 160 KB of ROM;
- 72 KB of EEPROM;
- 4KB of RAM;
- Embedded crypto-processor;
- 3xDES, AES, RSA, ECC;
- Certification according to Common Criteria (CC) EAL5+ level;
- Security Certificates from payment operators.

1.2 Grid Of Secure Elements (GoSE)

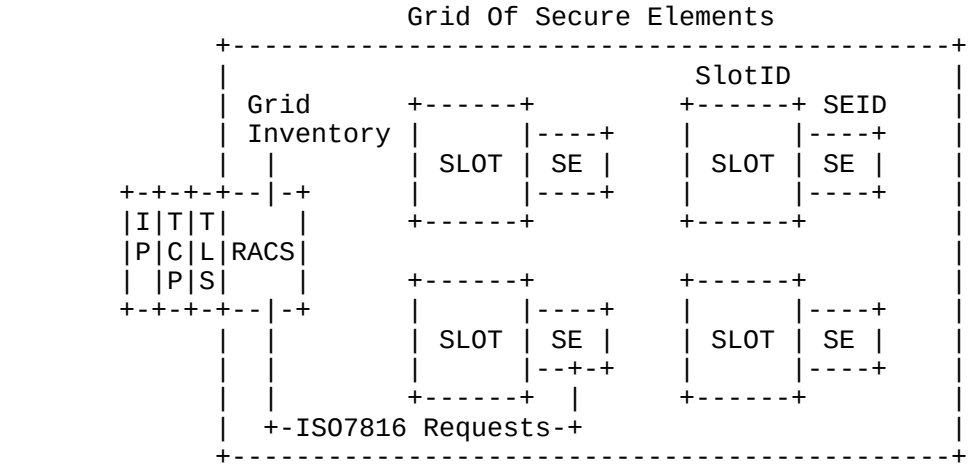

Figure 1. Architecture of a Grid of Secure Elements

A grid of Secure Elements (GoSE) is a server hosting a set of secure elements.

The goal of these platforms is to deliver trusted services over the Internet. These services are available in two functional planes,
- The user plane, which provides trusted computing and secure storage.
- The management plane, which manages the lifecycle (downloading, activation, deletion) of applications hosted by the Secure Element.

A grid of Secure Elements offers services similar to HSM (Hardware Secure Module), but may be managed by a plurality of administrators, dealing with specific secure microcontrollers.

According to this draft all accesses to a GoSE require the TCP transport and are secured by the TLS [TLS 1.0] [TLS 1.1] [TLS 2.0] protocol.

The RACS protocol provides all the features needed for the remote use of secure elements, i.e.
- Inventory of secure elements
- Information exchange with the secure elements

1.3 Secure Element Identifier (SEID)

Every secure element needs a physical slot that provides power feeding and communication resources. This electrical interface is for example realized by a socket soldered on an electronic board, or a CAD (Card Acceptance Device, i.e. a reader) supporting host buses such as USB.

Within a GoSE each slot is identified by a SlotID (slot identifier) attribute, which may be a socket number or a CAD name.

The SEID (Secure Element Identifier) is a unique identifier indicating that a given SE is hosted by a GoSE. It also implicitly refers the physical slot (SlotID) to which the SE is plugged.

The GoSE manages an internal table that establishes the relationship between SlotIDs and SEIDs.

Therefore three parameters are needed for remote communication with secure element, the IP address of the GoSE, the associated TCP port, and the SEID.

1.4 APDUs

According to the [ISO7816] standards secure element process ISO7816 request messages and return ISO7816 response messages, named APDUs (application protocol data unit).

1.4.1 ISO7816 APDU request

An APDU request comprises two parts: a header and an optional body.

The header is a set of four or five bytes noted CLA INS P1 P2 P3

- CLA indicates the class of the request, and is usually bound to standardization committee (00 for example means ISO request).
-INS indicates the type of request, for example B0 for reading or D0 for writing.
- P1 P2 gives additional information for the request (such index in a file or identifier of cryptographic procedures)
- P3 indicates the length of the request body (from P3=01 to P3=FF), or the size of the expected response body (a null value meaning 256 bytes). Short ISO7816 requests may comprise only 4 bytes

- The body may be empty. Its maximum size is 255 bytes

1.4.2 ISO7816 APDU response

An APDU response comprises two parts an optional body and a mandatory status word.

- The optional body is made of 256 bytes at the most.

- The response ends by a two byte status noted SW. SW1 refers the most significant byte and SW2 the less significant byte.

An error free operation is usually associated to the 9000 status word. Following are some interpretations of the tuple SW1, SW2 according to various standards:

- ’9F’ ’xx’, indicates that xx bytes (modulus 256) are ready for reading. Operation result MUST be fetched by the ISO Get Response APDU (CLA = ’C0’, P1=P2= ’0’, P3= ’XX’)
- ’67’ ’XX’, incorrect parameter P3
- ’6B’ ’XX’, incorrect parameter P1 or P2
- ’6D’ ’XX’, unknown instruction code (INS) given in the request
- ’6E’ ’XX’, wrong instruction class (CLA) given in the request
- ’6F’ ’XX’, technical problem, not implemented...
- ’61 ’’XX’, operation result MUST be fetched by the ISO Get Response APDU (CLA = ’C0’, P1=P2= ’0’, P3= ’XX’)
- ’6C ’’XX’, operation must be performed again, with the LE parameter value sets to ’XX’.

2 The RACS protocol

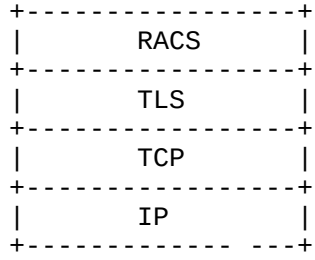

Figure 2. The RACS stack

The RACS protocol works over the TCP transport layer and is secured by the TLS protocol. The TLS client (i.e. the RACS client) MUST be authenticated by a certificate.

One of the main targets of the RACS protocol is to efficiently push a set of ISO7816 requests towards a secure element in order to perform cryptographic operations in the user’s plane. In that case a

RACS request typically comprises a prefix made with multiple ISO7816 requests and a suffix that collects the result of a cryptographic procedure.

The use of TLS with mutual authentication based on certificate provides a simple and elegant way to establish the credentials of a RACS client over the GoSE. It also enables an easy splitting between users' and administrators' privileges.

2.1 Structure of RACS request

A RACS request is a set of command lines, encoded according to the ASCII format. Each line ends by the CR (carriage Return) and line feed (LF) characters.

Each command is a set of tokens (i.e. words) separated by the space (0x20) character(s).

The first token of each line is the command to be executed.

A command line MAY comprise other tokens, which are called the command parameters.

A RACS request always starts by a BEGIN command and ends by an END command.

The processing of a RACS request is halted after the first error. In that case the returned response contained the error status induced by the last executed command.

2.2 Structure of a RACS response

A RACS response is a set of lines, encoded according to the ASCII format. Each line ends by the CR (carriage Return) and line feed (LF) characters.

Each line is a set of tokens (i.e. words) separated by the space (0x20) character(s).

The first token of each line is the header.

A response line MAY comprise other tokens, which are called the response parameters.

Three classes of headers are defined BEGIN, END and Status.

A RACS response always starts by a BEGIN header and ends by an END header. It comprises one status line.

2.2.1 BEGIN Header

This header starts a response message.

It comprises an optional parameter, an identifier associated to a previous request message.

2.2.2 END Header

This header ends a response message.

2.2.3 Status line

A status header indicates a status line. It begins by the character '+' in case of success or '-' if an error occurred during the RACS request execution. It is followed by an ASCII encoded integer, which is the value of the status.

A status line MAY comprise other tokens, which are called the response parameters.

2.2.4 Examples of RACS responses:

```
BEGIN
+000 Success
END

BEGIN Asterix237
+000 [Hexadecimal Encoded ISO7816-Response]
END

BEGIN
-400 Unknown command at line 2
END

BEGIN moon1969
-500 Conditions not satisfied at line 7
END

BEGIN
-600 Timeout occurred at line 2
END
```

2.3 RACS request commands

2.3.1 BEGIN

This command starts a request message. A response message is returned if an error is detected.

An optional parameter is the request identifier, which MUST be echoed in the parameter of the first response line (i.e. starting by the BEGIN header).

2.3.2 END

This command ends a request message. It returns the response message triggered by the last command.

Example1
========
Request:
BEGIN CR LF
END CR LF
Response:
BEGIN CR LF
+000 Success CR LF
END CR LF

Example2
========
Request:
BEGIN Marignan1515 CR LF
APDU ASTERIX-CRYPTO-MODULE [ISO7816-Request] CR LF
END CR LF

Response:
BEGIN Marignan1515 CR LF
+000 [ISO7816-Response] CR LF
END CR LF

2.3.3 GET-VERSION

This command requests the current version of the RACS protocol. The returned response is the current version encoded by two integer separated by the '.' character. The first integer indicates the major version and the second integer gives the minor version.

Example
=======
Request:
BEGIN CR LF
GET-VERSION CR LF
END CR LF

```
Response:
BEGIN CR LF
+000 1.0 CR LF
END CR LF
```

2.3.4 SET-VERSION

This command sets the version to be used for the RACS request. An error status is returned by the response if an error occurred.

```
Example 1
=========
Request:
BEGIN CR LF
SET-VERSION 2.0 CR LF
END CR LF

Response:
BEGIN CR LF
-400 Error line 2 RACS 2.0 is not supported
END CR LF

Example 2
=========
Request:
BEGIN CR LF
SET-VERSION 1.0 CR LF
END CR LF

Response:
BEGIN CR LF
+000 RACS 1.0 has been activated CR LF
END CR LF
```

2.3.5 LIST

This command requests the list of SEID plugged in the GoSE.

It returns a list of SEIDs separated by space (0x20) character(s).

Some SEID attributes could be built from a prefix and an integer suffix (such as SE#100 in which SE# is the suffix and 100 is the integer suffix. A list of non-consecutive SEID could be encoded as prefix[i1;i2;..;ip] where i1,i2,ip indicates the integer suffix. A list of consecutive SEID could be encoded as prefix[i1-ip] where i1,i2,ip indicates the integer suffix.

```
Example 1
=========
```

```
Request:
BEGIN CR LF
LIST CR LF
END CR LF

Response:
BEGIN CR LF
+000 SEID1 SEID2 CR LF
END CR LF

Example 2
=========
Request:
BEGIN CR LF
LIST CR LF
END CR LF

Response:
BEGIN CR LF
+000 device[1000-2000] serialnumber[567;789;243] CR LF
END CR LF
```

2.3.6 RESET

This command resets a secure element. The first parameter gives the secure element identifier (SEID). An optional second parameter specifies a warm reset. The default behavior is a cold reset. The response status indicates the success or the failure of the operation.

```
Syntax: RESET SEID [WARM] CR LF

Example 1
=========
Request:
BEGIN CR LF
RESET device#45 CR LF
END CR LF

Response:
BEGIN CR LF
+000 device#45 Reset Done
END CR LF


Example 2
=========
Request:
BEGIN CR LF
RESET device#45 WARM CR LF
END CR LF
```

Response:
BEGIN CR LF
+000 device#45 Warm Reset Done
END CR LF

2.3.7 APDU

This command sends an ISO7816 request to a secure element or a set of ISO7816 commands.

The first parameter specified the SEID

The second parameter is an ISO7816 request

Three optional parameters are available; they MUST be located after the second parameter.

- CONTINUE=value, indicates that the next RACS command will be executed only if the ISO7816 status word (SW) is equal to a given value. Otherwise an error status is returned.
- MORE=value, indicates that a FETCH ISO7816 request will be performed (i.e. a new ISO7816 request will be sent) if the first byte of the ISO7816 status word (SW1) is equal to a given value.
- FETCH=value fixes the four bytes of the ISO7816 FETCH request (i.e. CLA INS P1 P2). The default value is 00C00000 (CLA=00, INS=C0, P1=00, P2=00)

When the options CONTINUE and MORE are simultaneously set the SW1 byte is first checked. If there is no match then the SW word is afterwards checked.

SYNTAX
APDU SEID ISO7816-REQUEST [CONTINUE=SW] [MORE=SW1] [FETCH=CMD]

The returned response is the ISO7816 response. If multiple ISO7816 requests are executed (due to the MORE option), the bodies are concatenated in the response, which ends by the last ISO7816 status word.

Example 1
=========
Request:
BEGIN CR LF
APDU SEID ISO7816-REQUEST CR LF
END CR LF

Response:
BEGIN CR LF
+000 ISO7816-RESPONSE CR LF
END CR LF

```
Example 2
=========
BEGIN CR LF
APDU SEID ISO7816-REQUEST-1 CONTINUE=9000 CR LF
APDU SEID ISO7816-REQUEST-2 CR LF
END CR LF

Response:
BEGIN CR LF
+000 ISO7816-RESPONSE-2 CR LF
END CR LF

Example 2
=========
BEGIN CR LF
APDU SEID ISO7816-REQUEST-1 CONTINUE=9000 CR LF
APDU SEID ISO7816-REQUEST-2 CR LF
END CR LF

Response:
BEGIN CR LF
-300 Request Error line 1 wrong SW CR LF
END CR LF

Example 3
=========
BEGIN CR LF
APDU SEID ISO7816-REQ-1 CONTINUE=9000 CR LF
APDU SEID ISO7816-REQ-2 CONTINUE=9000 CR LF
APDU SEID ISO7816-REQ-3 CONTINUE=9000 MORE=61 FETCH=00C00000 CR LF
END CR LF

Response:
BEGIN CR LF
+000 ISO7816-RESP-3 CR LF
END CR LF

Multiple ISO7816 requests have been performed by the third APDU
command according to the following scenario
- the ISO7816-REQ-3 request has been forwarded to the secure element
(SEID)
- the ISO 7816 response comprises a body (body0) and a status word
(SW0) whose first byte is 0x61
- the FETCH command CLA=00, INS=00, P1=00, P2=00, P3=SW2 is sent to
the secure element
- the ISO 7816 response comprises a body (body1) and a status word
(SW1) set to 9000

The RACS response is set to
+000 body0 || body1 || SW1 CR LF
where ||indicates a concatenation operation.
```

3 URI for the GoSE

The URI addressing the resources hosted by the GoSE is represented by the string:

RACS://GoSE-Name:port/?request

where request is the RACS request to be forwarded to a the GoSE.

RACS command lines are encoded in a way similar to the INPUT field of an HTML form. Each command is associated to an INPUT name, the remaining of the command line i.e. a set of ASCII characters, is written according to the URL encoding rules. End of line characters, i.e. carriage return (CR) and line feed (LF) are omitted.

As a consequence a request is written to the following syntax cmd1=cmd1-parameters&cmd2=cmd2-parameters

Example:
RACS://GoSE-Name:port/?BEGIN=&APDU=SEID%20[ISO7816-REQUEST]&END=

4 HTTP interface

A GoSE SHOULD support an HTTP interface. RACS requests/responses are transported by HTTP messages. The use of TLS is mandatory.

4.1 HTTPS Request

https://GoSE-Name:port/RACS?request

where request is the RACS request to be forwarded to a secure element (SEID)

The RACS request is associated to an HTML form whose name is "RACS". The request command lines are encoded as the INPUT field of an HTML form. Each command is associated to an INPUT name, the remaining of the command line i.e. a set of ASCII characters is written according to the URL encoding rules. End of line characters, i.e. carriage return (CR) and line feed (LF) are omitted.

As a consequence a RACS request is written as https://GoSE-Name/RACS?cmd1=cmd1-parameters&cmd2=cmd2-parameters

Example:

https://GoSE-Name/RACS?BEGIN=&APDU=SEID%20[ISO7816-REQUEST]&END=

4.2 HTTPS response

The RACS response is returned in an XML document.

The root element of the document is <RACS-Response>

The optional parameter of the BEGIN header, is the content of the <begin> element.

The status header is the content of the <status> element.

The parameters of the status line are the content of the <parameters> element.

The END header is associated to the element <end>

End of line, i.e. carriage return (CR) and line feed (LF) characters are omitted.

As a consequence a RACS response is written as :
<RACS-Response>
<begin>Optionnal-ID</begin>
<status>+000</status>
<parameters>parameters of the RACS response</parameters>
<end></end>
</RACS-Response

5 Security Considerations

To be done.

6 IANA Considerations

7 References

7.1 Normative References

[TLS 1.0] Dierks, T., C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999

[TLS 1.1] Dierks, T., Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006

[TLS 1.2] Dierks, T., Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.1", draft-ietf-tls-rfc4346-bis-10.txt, March 2008

[ISO7816] ISO 7816, "Cards Identification - Integrated Circuit Cards with Contacts", The International Organization for Standardization (ISO)

7.2 Informative References

[REST} Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", 2000, http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[GP] Global Platform Standards, http://www.globalplatform.org

[EUROSMART] The EUROSMART association, http://www.eurosmart.com

8 Authors' Addresses

Pascal Urien
Telecom ParisTech
23 avenue d'Italie
75013 Paris Phone: NA
France Email: Pascal.Urien@telecom-paristech.fr