HIP support for RFIDs
draft-irtf-hiprg-rfid-03

Abstract

   This document describes an architecture based on the Host Identity
   Protocol (HIP), for active RFIDs, i.e. Radio Frequency Identifiers
   including tamper resistant computing resources, as specified for
   example in the ISO 14443 or 15693 standards. HIP-RFIDs never expose
   their identity in clear text, but hide this value (typically an EPC-
   Code) by a particular equation that can be only solved by a dedicated
   entity, referred as the portal. HIP exchanges occur between HIP-RFIDs
   and portals; they are transported by IP packets, through the Internet
   cloud.

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 [RFC2119].

Status of this Memo

Copyright Notice

Table of Contents

1 Overview

1.1 Motivation

   RFIDs are electronic devices, associated to things or computers,
   which transmit their identifier (usually a serial number) via radio
   links. The Host Identity Protocol [HIP] is a security protocol based
   on the use of cryptographic identifiers, and specified for IP-based
   networks [HIP].

   The first motivation for designing HIP support for RFIDs is to
   enforce a strong privacy for the Internet of Things, e.g. identity is
   protected by cryptographic procedures compatible with RFID computing
   resources. As an illustration, EPC codes or IP addresses are today
   transmitted in the clear.

   The second motivation is to define an identity layer for RFIDs
   logically independent from the transport facilities, which may
   optionally support IP stacks.

   In other words, we believe that the Internet of Things will be
   Identity oriented; RFIDs will act as electronic ID for objects to
   which they are linked. In this context, privacy is a major challenge.

1.2 Passive and active RFIDs

   An RFID is a slice of silicon whose area is about 1 mm2 for
   components used as cheap electronic RFIDs, and around 25 mm2 for
   chips like contact-less smart cards inserted in passports and mobile
   phones.

   RFIDs are divided into two classes, the first includes devices that
   embed CPU and memory (RAM, ROM, E2PROM) such as contact-less smart
   cards, and the second comprises electronic chips based on cabled
   logic circuits.

   There are multiple standards relative to RFIDs. The ISO 14443
   standard introduces components dealing with the 13.56 MHz frequency
   that embed a CPU and consume about 10mW; data throughput is about 100
   Kbits/s and the maximum working distance (from the reader) is around
   10cm.

   The ISO 15693 standard also uses the same 13.56 MHz frequency, but
   enables working distances as high as one meter, with a data
   throughput of a few Kbits/s.

   The ISO 18000 standard defines parameters for air interface
   communications associated with frequency such as 135 KHz, 13.56 MHz,
   2.45 GHz, 5.8 GHz, 860 to 960 MHz and 433 MHz. The ISO 18000-6
   standard uses the 860-960 MHz range and is the basis for the Class-1

    Generation-2 UHF RFID, introduced by the EPCglobal [EPCGLOBAL]
    consortium.

1.3 About the Internet of Things (IoT)

    The term "Internet of Thing (IoT)" was invented by the MIT Auto-ID
    Center, in 2001, and refers to an architecture that comprises four
    levels,

    - Passive RFIDs, such as Class-1 Generation-2 UHF RFIDs, introduced
    by the EPC Global consortium and operating in the 860-960 MHz range.

    - Readers plugged to a local (computing) system, which read the
    Electronic Product Code [EPC].

    - A local system, offering IP connectivity, which collects
    information pointed by the EPC thanks to a protocol called Object
    Naming Service (ONS)

    - EPCIS (EPC Information Services) servers, which process incoming
    ONS requests and returns PML (Physical Markup Language) files [PML],
    e.g. XML documents that carry meaningful information linked to RFIDs.

1.4 HIP-RFIDs

```
             PORTAL                    READER            RFID


    +----------------------+                        +-----------+
    !                      !                        ! +-------+ !
    !             +-----+ !<=======================>! +  HIP  + !
    !  +---------+ + HIP + !    +------------------+  ! +-------+ !
    !  + IDENTITY+ +-----+ !    +------------------+  ! +-------+ !
    !  + SOLVER  +   [HEP] !<=>! [HEP]            !  ! !   |    ! !
    !  +---------+ +-----+ !    ! +------+------+  !  ! +-------+ !
    !             +     + !    ! +      + RFID + !  ! + RFID  + !
    !    EPC-Code  + IP + !<=>! +  IP  + Radio + !<>! + Radio + !
    !             +     + !    ! +      + Ptcol + !  ! + Ptcol + !
    !             +-----+ !    ! +------+------+  !  ! +-------+ !
    !                      !    !                  !  !         !
    +----------+-----------+    +------------------+  +-----------+
              !
              V
       TO EPC GLOBAL
         SERVICES
```
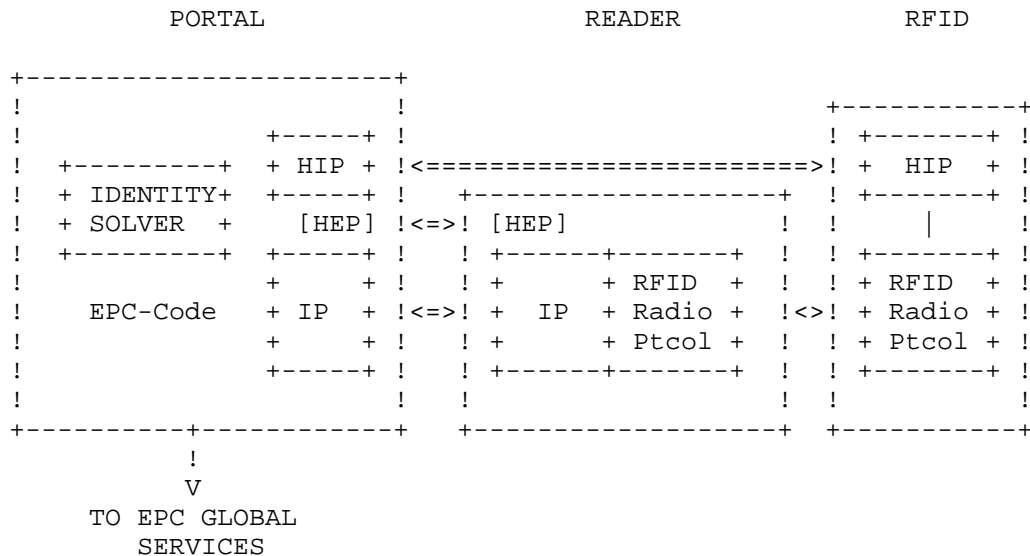
    Figure 1. HIP-RFID Architecture

    This document suggests embedding a modified version of a HIP-enabled
    stack in active RFIDs, named HIP-RFIDs. It assumes that such devices
    would not support an IP stack, but should be rather identity
    oriented, i.e. will use readers' IP resources in order to unveil

their EPC-Code only to trusted entities (called portals in the
architecture shown by Figure 1). Privacy, e.g. identity protection
seems a key prerequisite [SEC] before the effective massive
deployment of these devices.

The HIP-RFID architecture includes three functional entities: HIP
RFIDs, RFID readers, and portals, and defines a new HIP encapsulation
protocol (HEP):

- HIP RFIDs. HIP, as defined in [HIP], is transported by IP packets.
HIP-RFIDs support a modified version of this protocol but do not
require end-to-end IP transport.

- RFID readers. These provide IP connectivity and communicate with
RFIDs through radio links either defined by EPC Global or ISO
standards. The IP layer transports HIP messages between RFIDs and
other HIP entities. According to HIP, an SPI (Security Parameter
Index) associated to an IPsec tunnel MAY be used by the IP host (e.g.
a reader) in order to route HIP packets to/from the right software
identity.

- HEP, HIP Encapsulation Protocol. HIP messages MAY be encapsulated
by protocols such as UDP or TCP in order to facilitate HIP transport
in existing software and networking architectures. The HEP does not
modify the content of an HIP packet. This class of protocol is not
specified by this document.

- PORTAL entity. This device manages a set of readers; it is a HIP
entity that includes a full IP stack. Communications between portal
and RFIDs logically work as peer to peer HIP exchanges. RFID
identifier (HIT) is hidden and appears as a pseudo random value;
within the portal a software block called the IDENTITY SOLVER
resolves an equation f, whose solution is an EPC Code. The portal
accesses EPCIS services; when required privacy may be enforced by
legacy protocol such as SSL or IPsec.

- The portal maintains a table linking HIT and EPC-Code. It acts as a
router for that purpose it MUST provide an identity resolution
mechanism, i.e. a relation between HIT and EPC-Code.

1.5 Main differences between HIP-RFID and HIP

In HIP [HIP], the HIT (Host Identifier Tag) is a fixed value obtained
from the hash of an RSA public key. This parameter is therefore
linked to a unique identity, and can be used for traceability
purposes; in other words HIP does not natively include privacy
features.

In [BLIND], it is proposed to hide the HIT with a random number
thanks to a hash function, i.e.

B-HIT = sha1(HIT || N), with N a random value and || the
concatenation operation.

The case in which only one HIT (either initiator or responder) is
blinded looks similar to the HIP-RFID protocol described in this
draft working with a particular transform (HMAC Transform, 0x0001).

2. Basic Exchange

The HIP-RFID base exchange (T-BEX) is derived from the "classical"
base exchange (BEX), introduced in [HIP]. It is a four way handshake
illustrated by Figure 2.

```
  RFID               READER                        PORTAL
  --+--              --+--                         ---+---
   !      START       !                              !
   !<---------------!                                !
   !                  !                              !
   !  I1-T                                           !
   !  HIT-I  HIT-R                                   !
   ! ------------------------------------------------> !
   !                                                 !
   !                                                 !
   !  R1-T                                           !
   !  HIT-I  HIT-R  R-T(r1) HIP-T-Transforms         !
   !  [*ESP-Transforms]                              !
   ! <------------------------------------------------ !
   !                                                 !
   !                                                 !
   !  I2-T                                           !
   !  HIT-I HIT-R HIP-T-Transform [*ESP-Transform] R-T(r2) !
   !  F-T=f(r1, r2, EPC-Code) [*ESP-Info] MAC-T      !
   ! ------------------------------------------------> !
   !                                                 !
   !                                                 !
   !  R2-T                                           !
   !  HIT-I HIT-R  [*ESP-Info]  MAC-T                !
   ! <------------------------------------------------ !
   !                                                 !
   !                                                 !
   !               Optional ESP Dialog               !
   ! <------------------------------------------------> !
   !                                                 !
   !                                                 !
```

Figure 2. HIP-RFIDs Base Exchange (T-BEX), *means optional attributes

A HEP layer MAY be used to transport HIP messages in a non-IP
context, but this optional facility is out of scope for this
document.

2.1 I1-T

   When a reader detects an RFID, it realizes all low level operations
   in order to set up a radio communication link. Finally the reader
   delivers a START message that triggers the RFID.

   The HIP-RFID sends the I1-T packet (I suffix meaning initiator), in
   which HIT-I is a pseudorandom value internally generated by the HIP-
   RFID.

   If the RFID doesn't known the portal HIT it sets the HIT-R value to
   zero; in that case the reader MAY modify this field in order to
   identify the appropriate entity.

   The I1-T message is not MACed.

2.2 R1-T

   The portal produces the R1-T (R suffix meaning responder) packet,
   which includes a nonce r1 and optional parameters. These fields
   indicate a list of supported authentication schemes (HIP-T-
   TRANSFORMs) and a list of ESP-TRANSFORMs, i.e. secure channels that
   could be opened between portal and RFIDs.

   This message includes the following fields:
   - HIT-I, a random number which identifies a RFID
   - HIT-R, the portal HIP, either a null or fixed value.
   - HIT-T-TRANSFORMs, a list of authentication schemes
   - ESP-T-TRANSFORMs, an optional list of ESP secure channels

   The R1-T message is not MACed.

2.3 I2-T

   The HIP-RFID builds the I2-T message, which contains

   - The selected HIP-T-TRANSFORM (the current authentication scheme).
   - An optional ESP-TRANSFORM (a class of secure channel between RFID
   and portal).
   - A nonce r2, included in the R-T attribute.
   - An equation f(r1, r2, EPC-Code), whose solution, according to the
   selected HIP-T-TRANSFORM, unveils the EPC-Code value.
   - An optional ESP-Info attribute that gives information about the
   secure (ESP) channel, and which includes the SPI-I value.
   - A keyed MAC (MAC-T), which works with a KI-Auth-key deduced from
   r1, r2 and the hidden EPC-Code value.

   KI-Auth-key = g(r1, r2, EPC-Code)

   The keyed MAC is computed over the complete I2-T message, the content
   of MAC-T resulting from this calculation is initially set to a null
   value.

   The portal and the RFID shares secret keys. The meaning of these keys
   are dependent upon the f equation.

   In some cases the EPC-Code is the only shared key. The portal knows a
   list of EPC-Code and tries all solutions for solving f, according to
   brute force techniques. As an illustration a hash function may be
   used for f:

   f= sha1(r1 || r2 || EPC-Code), where || is the concatenation
   operation.

   In other cases a set of keys is shared between portal and RFIDs. For
   example a binary tree of HMAC procedure MAY be used, each HMAC beeing
   associated to a particular key. A binary tree of depth n may identify
   2**n RFIDs, each of them stores n keys (ki:j). The f function is a
   list of n values such as

                        HMAC(r1 || r2, ki:j)

   Where ki:j is a secret key, and j the bit value (either 0 or 1) at
   the rank i (ranging between 0 and n-1) for the EPC-Code (or RFID
   index).

2.4 R2-T

   The fourth and last R2-T packet is optional. It includes

   - A keyed MAC (MAC-T) computed with the KI-Auth-key deduced from r1,
   r2 and the hidden EPC-Code value.

   KI-Auth-key = g(r1, r2, EPC-Code)

   - An optional ESP-Info attribute that gives information about the
   secure (ESP) channel, and which includes the SPI-R value.

   The R2-T packet is mandatory when an ESP channel has been previously
   negotiated. ESP channel is required if the portal intends to perform
   read or write operations with the RFIDs.

2.5 HIT format

   HIT-R MAY be a fixed value embedded in the RFID during the
   manufacturing process or a null value if no specific portal is
   required.

   HIT-I MAY comprise an optional header given coded according to
   various hierarchical rules and MUST include a trailer, which is a
   true random number.

2.6 State Machine

   The state machine is similar the one described in [RFC 5201]. No
   retry operations are performed, because the communication with the
   RFID may be lost at any time. Furthermore RFIDs are generally not
   equipped with timers.

   2.6.1 Unassociated.

   The state machine starts.

   2.6.2 I1-Sent

   The RFID has been reset by the reader, and has sent the I1-T message.

   2.6.3 R1-Sent

   The responder has received the I1-T message and has sent the R1-T
   packet.

   2.6.4 I2-Sent

   The RFID has received the R1-T packet, and has sent the I2-T message.

   2.6.5 R2-Sent

   The responder has received the I2-T message and has sent the optional
   R2-T packet.

   2.6.6 Established

   The RFID has received the R2-T message. A secure channel is
   established.

3. Formats

3.1 Payload

   The payload format is imported from the [HIP] specification.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Next Header   | Header Length |0| Packet Type | VER. | RES.|1|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Checksum             |           Controls           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |               Sender's Host Identity RFID (HIT)               |
   |                                                              |
   |                                                              |
   |                                                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              Receiver's Host Identity RFID (HIT)             |
   |                                                              |
   |                                                              |
   |                                                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                              |
   /                      HIP Parameters                          /
   /                                                              /
   |                                                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
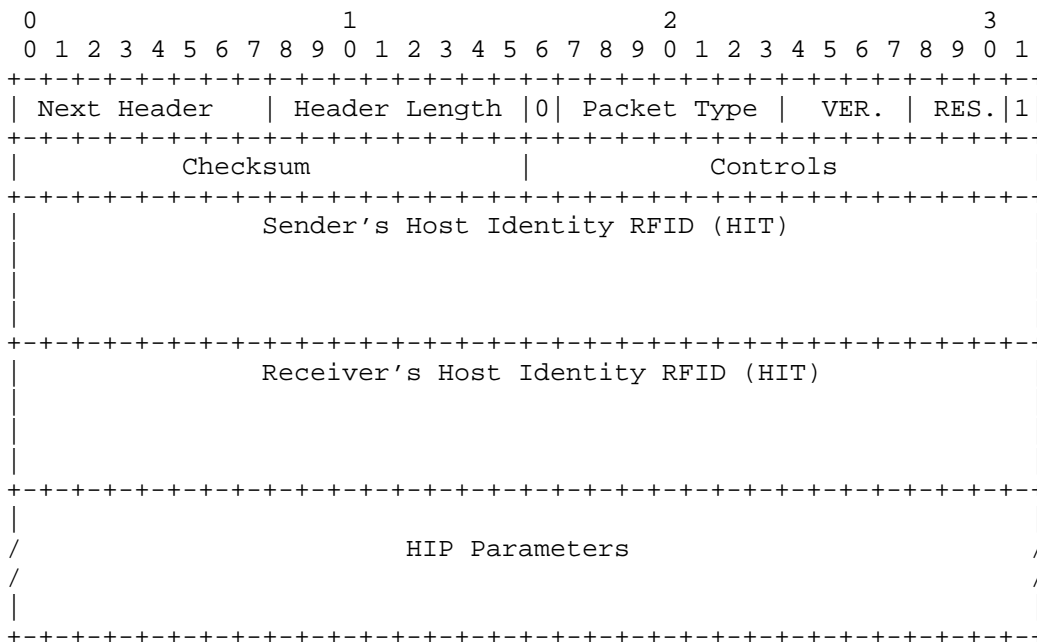
   Next Header : normal value is decimal 59, IPPROTO_NONE.

   Header Length: the length of the HIP Header and HIP parameters in 8
   bytes units, excluding the first 8 bytes

   Packet Type: Detailed in section 4.2

   VER: 0001

   RES: 000

   Checksum: This checksum covers the source and destination addresses
   in the IP header.

   HIP-RFIDs always deliver HIP packets with the null value for the
   checksum field. The reader MUST compute the checksum.

   HIP-RFIDs do not check the checksum of received packets.

   Controls: this field is reserved for future use (RFU)

   Sender's Host Identity RFID: 16 bytes HIT

Receiver's Host Identity RFID: 16 bytes HIT

HIP Parameters: a list of attributes encoded in the TLV format

3.2 Packet types

```
+----------------+-------------------------------------------+
|  Packet type   | Packet name                               |
+----------------+-------------------------------------------+
|     0x40       | I1-T - The HIP-RFID Initiator Packet      |
|                |                                           |
|     0x41       | R1-T - The HIP-RFID Responder Packet      |
|                |                                           |
|     0x42       | I2-T - The Second HIP-RFID Initiator Packet|
|                |                                           |
|     0x43       | R2-T - The Second HIP-RFID Responder Packet|
|                |                                           |
+----------------+-------------------------------------------+
```
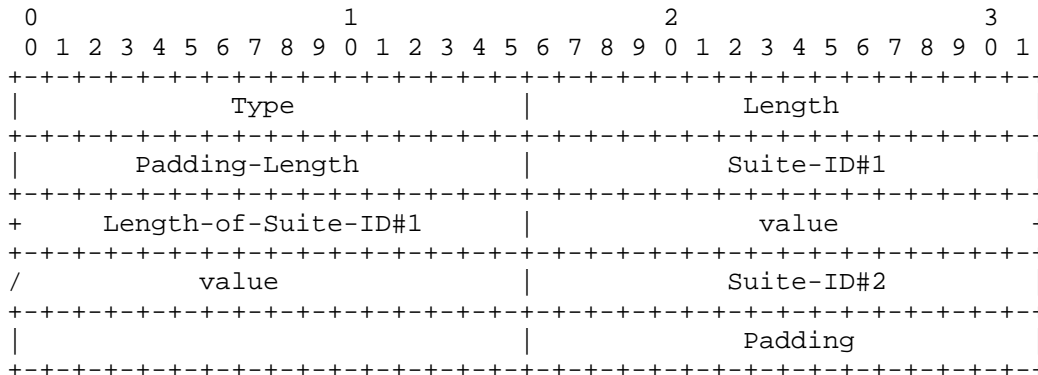
3.3 Summary of HIP parameters

```
+----------------------+-------+----------+----------------------+
| TLV                  | Type  | Length   | Data                 |
+----------------------+-------+----------+----------------------+
| R-T                  | 0x400 | variable | Random value r1 or r2 |
|                      |       |          |                      |
| HIP-T-TRANSFORM      | 0x402 | variable | HIP-RFID transform(s) |
|                      |       |          |                      |
| F-T                  | 0x404 | variable | f function value     |
|                      |       |          |                      |
| MAC-T                | 0x406 | variable | Keyed MAC            |
|                      |       |          |                      |
| ESP-Transform        | 0x408 | variable | ESP transform(s)     |
|                      |       |          |                      |
| ESP-Info             | 0x40A | variable | ESP parameter(s)     |
|                      |       |          |                      |
+----------------------+-------+----------+----------------------+
```

3.4 R-T
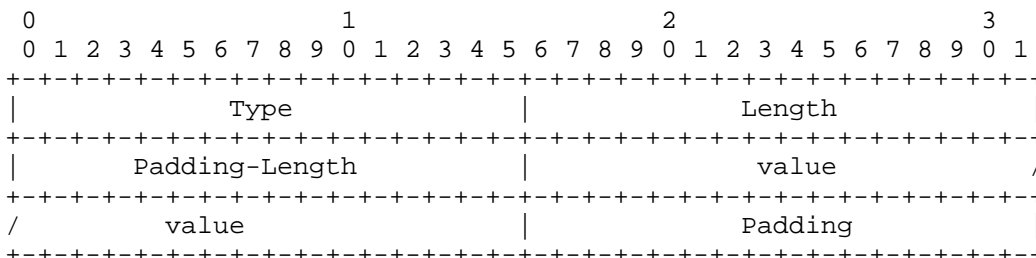
```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |            Type               |             Length           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |       Padding-Length          |             value           /
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /         value                 |           Padding            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Type            0x400
        Length          total length in bytes
        Value           random value
        Padding-Length  padding length in bytes
        Padding         padding bytes
```
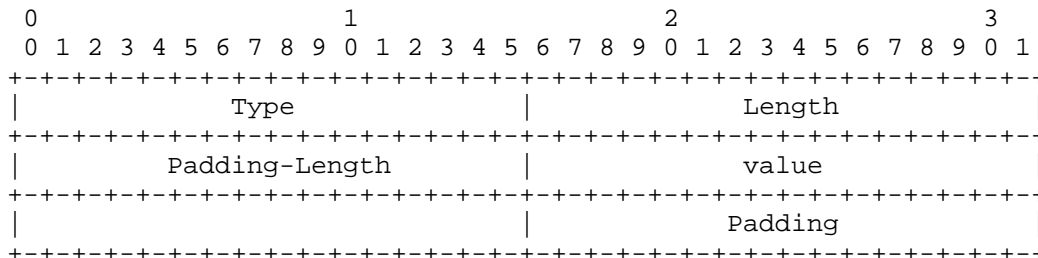
3.5 HIP-T-Transform

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              Type               |             Length            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |        Padding-Length           |           Suite-ID#1          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    +     Length-of-Suite-ID#1         |              value           +
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /          value                  |           Suite-ID#2          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                 |            Padding            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Type                 0x402
        Length               Total length
        Padding-Length       Number of padding bytes
        Suite-ID             Defines the HIP Cipher Suite to be used
        Length-of-Suite-ID   Defines the length of optional data
        Padding              Padding bytes
```
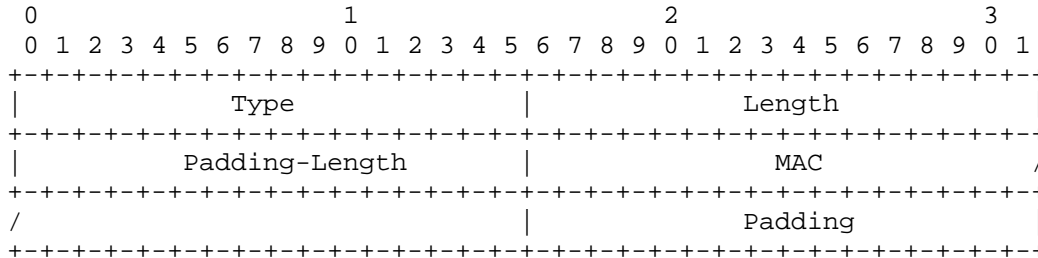
3.6 F-T

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              Type               |             Length            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Padding-Length          |              value           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                 |            Padding            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

        Type             0x404
        Length           total length, in bytes
        Padding-Length   padding length in bytes
        Value            the f value with a variable length
        Padding          padding bytes
```

3.7 MAC-T

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |             Type              |            Length             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Padding-Length        |            MAC                /
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    /                               |           Padding             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

        Type   0x406
        Length   total length, in bytes
        Padding-Length padding length, in bytes
        Value   Keyed MAC value
        Padding   padding bytes

   A MAC procedure works with the K-Auth-Key and is computed over the
   whole HIP message according to the following rules

   - The checksum field of the HIP header is set to a null value.

   - The MAC field of the MAC-T attribute is set to a null value

3.8 ESP-Transform

   Details of the attribute will be specified by another document.

3.9 ESP-Info

   Details of the attribute will be specified by another document.

4. BEX Example

4.1 Generic example

   4.1.1 I1-T

         Next Header: 0x3B
         Header Length: 0x4
         Packet Type: 0x40
         Version: 0x1
         Reserved: 0x1
         Control: 0x0
         Checksum: 0x0000
         Sender's HIT (RFID) :   0x0123456789ABCDEF
                                           0123456789ABCDEF
         Receiver's HIT (Portal) :   0x0000000000000000
                                           0000000000000000

  The checksum is computed by portal and reader according to rules
  specified in [HIP]; it covers the source and destination IP
  addresses.

   4.1.2 R1-T

         Next Header: 0x3B
         Header Length: 0xB
         Packet Type: 0x41
         Version: 0x1
         Reserved: 0x1
         Control: 0x0
         Checksum: 0xabcd
         Sender's HIT (Portal)   0xA5A5A5A5A5A5A5A5
                                           5A5A5A5A5A5A5A5A
         Receiver's HIT (RFID)   0x0123456789ABCDEF
                                           0123456789ABCDEF
         R-T   0x040000280002rrrr
                                           rrrrrrrrrrrrrrrr
                                           rrrrrrrrrrrrrrrr
                                           rrrrrrrrrrrrrrrr
                                           rrrrrrrrrrrrpppp
         HIP-T-Transforms   0x0402001000020001
                                           000000020000pppp

   r1 is a 128 bits value
   Transforms 1, 2 are supported by the reader.

4.1.3 I2-T

```
        Next Header:                0x3B
        Header Length:              0x14
        Packet Type:                0x42
        Version:                    0x1
        Reserved:                   0x1
        Control:                    0x0
        Checksum:                   0x0000
        Sender's HIT (RFID) :       0x0123456789ABCDEF
                                        0123456789ABCDEF
        Sender's HIT (Portal) :     0xA5A5A5A5A5A5A5A5
                                        5A5A5A5A5A5A5A5A
        HIP-T-Transform             0x0402001000060001
                                        0000pppppppppppp
        R-T                         0x040000280002rrrr
                                        rrrrrrrrrrrrrrrr
                                        rrrrrrrrrrrrrrrr
                                        rrrrrrrrrrrrrrrr
                                        rrrrrrrrrrrrpppp
        F-T                         0x040400280002ffff
                                        ffffffffffffffff
                                        ffffffffffffffff
                                        ffffffffffffffff
                                        ffffffffffffpppp
        MAC-T                       0x040600040006ssss
                                        ssssssssssssssss
                                        ssssssssssssssss
                                        sssspppppppppppp
```

The RFID selects the HIP-Transform number one. It produces an r2
nonce and computes a f value. It appends a 20 bytes keyed MAC.

    4.1.4 R2-T

          Next Header: 0x3B
          Header Length: 0x08
          Packet Type: 0x40
          Version: 0x1
          Reserved: 0x1
          Control: 0x0
          Checksum: 0xabcd
          Sender's HIT (RFID) :   0x0123456789ABCDEF
                                                0123456789ABCDEF
          Sender's HIT (Portal) :   0xA5A5A5A5A5A5A5A5
                                                5A5A5A5A5A5A5A5A
          MAC-T   0x040600040006ssss
                                                ssssssssssssssss
                                                ssssssssssssssss
                                                ssssppppppppppppp

    Reader ends the BEX-T.

4.2 HIP-T Transform 0x0001, HMAC

  EPC = 0123456789abcdefcdab

  4.2.1 I1-T

  << 3B 04 40 11 00 00 00 00 6A 68 2E 53 51 6B 51 6F
     2F 58 CE 60 25 42 1A E6 00 00 00 00 00 00 00 00
     00 00 00 00 00 00 00 00

  HEAD 3b044401100000000
  sHIT 6a682e53516b516f2f58ce6025421ae6
  dHIT 00000000000000000000000000000000

  4.2.2 R1-T

  >> 3B 0A 41 11 00 00 00 00 00 00 00 00 00 00 00 00
     00 00 00 00 00 00 00 00 6A 68 2E 53 51 6B 51 6F
     2F 58 CE 60 25 42 1A E6 04 00 00 20 00 06 27 6D
     03 4D DD 2D 52 79 3B 17 2C B9 5B CD 02 97 E2 DF
     61 15 00 00 00 00 00 00 04 02 00 10 00 06 00 02
     00 00 00 00 00 00 00 00

  HEAD 3b0a411100000000
  sHIT 00000000000000000000000000000000
  dHIT 6a682e53516b516f2f58ce6025421ae6

  ATT 0400 20 bytes   276d034ddd2d52793b172cb95bcd0297e2df6115
  ATT 0402 04 bytes   00020000

    4.2.3 I2-T

  << 3B 13 40 11 00 00 00 00 6A 68 2E 53 51 6B 51 6F
     2F 58 CE 60 25 42 1A E6 00 00 00 00 00 00 00 00
     00 00 00 00 00 00 00 00 04 02 00 10 00 06 00 01
     00 00 00 00 00 00 00 00 04 00 00 20 00 06 C5 95
     8B 23 6B 9B 0E AA 7A BB 25 F2 7D 24 C5 04 6E 89
     19 9E 00 00 00 00 00 04 04 00 20 00 06 80 1D
     BC 55 C5 F3 97 89 F8 3C 6C BA 14 50 18 7D 83 83
     3C AF 00 00 00 00 00 04 06 00 20 00 06 2A 23
     68 93 2B F7 3A BE C4 6B DD B8 3F 1B 3F 7F 9D ED
     8B 83 00 00 00 00 00 00

  HEAD 3b13401100000000
  sHIT 6a682e53516b516f2f58ce6025421ae6
  dHIT 00000000000000000000000000000000

  ATT 0402 04 bytes 00010000
  ATT 0400 20 bytes c5958b236b9b0eaa7abb25f27d24c5046e89199e
  ATT 0404 20 bytes 801dbc55c5f39789f83c6cba1450187d83833caf
  ATT 0406 20 bytes 2a2368932bf73abec46bddb83f1b3f7f9ded8b83

5. HIP-T-Transforms Definition

5.1 Type 0x0001, HMAC

  5.1.1 Suite-ID

  Suite-ID: 0x0001
  Length-of-Suite-ID: 0x0000

  5.1.2 F-T computing (f function)

  The F-T function produces a 20 bytes result, according to the
  relation:

  K = HMAC-SHA1(r1 | r2, EPC-Code)

  Y = f(r1, r2, EPC-Code) = HMAC-SHA1(K, CT1 | "Type 0001 key")

  Where:

  - SHA1 is the SHA1 digest function

  - EPC-Code is the RFID identity

  - HMAC-SHA1 is the keyed MAC algorithm based on the SHA1 digest
  procedure.

  - CT1 is a 32 bits string, whose value is equal to 0x00000001

- r1 and r2 are the two random values exchanged by the BEX

5.1.3 K-Auth-Key computing (g function)

The K-Auth-Key is computing according to the relation:

 K = HMAC-SHA1(r1 | r2, EPC-Code)

 Y = HMAC-SHA1(K, CT2 | "Type 0001 key")

 Where:

 - SHA1 is the SHA1 digest function

 - EPC-Code is the RFID identity

 - HMAC-SHA1 is the keyed MAC algorithm based on the SHA1 digest
 procedure.

 - CT2 is a 32 bits string, whose value is equal to 0x00000002

 - r1 and r2 are the two random values exchanged by the BEX

 5.1.4 MAC-T computing

 The HMAC-SHA1 function is used with the K-Auth-Key secret value:

 MAC-T(HIT-T packet) = HMAC-SHA1(K-Auth-Key, HIP-T packet)

5.2 Type 0x0002, Keys-Tree

 5.2.1 Suite-ID

 Suite-ID: 0x0002
 Length-of-Suite-ID: 0x0006
 Value1: an index identifying a HASH function (H), which produces t
 bytes.
 Value2: n, the depth of the tree, a two bytes number.
 Value3: p, the maximum number of child nodes, for each node, a two
 bytes number.

 The maximum elements of a keys-tree is therefore $p^{**}n$

 5.2.2 F-T computing (f function)

 The F-T function produces a list of Hi, 1<= i <= n, of nh bytes
 results, according to the relation:

 Y = f(r1, r2, EPC-Code) = H1 | H2 | Hi | Hn

   With
   Hi = HMAC-SHA1(r1 | r2, Ki:j)

   Where:

   - H is digest function producing t bytes

   - Ki:j is a set of pn secret keys.

   Each EPC-Code is associated with an index, whose value is written as:

                 RFID-Index = an p**(n-1) + an-1 p**(n-2) +   a1

   Each ai digit(ai p**(i-1))whose value ranges between 0 and p-1, is
   associated with a key Ki:j (i.e. the tree is made with pn keys, but
   only n values are stored in a given RFID), with j=ai

   - HMAC-H is the keyed MAC algorithm based on the H digest procedure.

   - r1 and r2 are the two random values exchanged by the BEX.

 5.2.3 K-Auth-Key computing (g function)

   The K-Auth-Key is computing according to the relation:

   K-Auth-Key = HMAC-H(r1 | r2, RFID-Index)

   Where:

   - H is a digest function producing t bytes

   - HMAC-H is the keyed MAC algorithm based on the H digest procedure.

   - RFID INDEX is the RFID index.

   - r1 and r2 are the two random values exchanged by the BEX.

 5.2.4 MAC-T computing

   The HMAC-H function is used with the K-Auth-Key secret value:

   MAC-T(HIT-T packet) = HMAC-H(K-Auth-Key, HIP-T packet)

6. Security Considerations

   To be done.

7. IANA Considerations

   To be done.

8 References

8.1 Normative references

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
   Requirement Levels", BCP 14, RFC 2119, March 1997.

   [HIP] R. Moskowitz, P. Nikander, P. Jokela, T. Henderson, Host
   Identity Protocol, RFC 5201, April 2008.

8.2 Informative references

   [EPC] Brock, D.L, The Electronic Product Code (EPC), A Naming Scheme
   for Physical Objects, MIT AUTO-ID CENTER, 2001.

   [PML] Brock, D.L - The Physical Markup Language, MIT AUTO-ID CENTER,
   2001.

   [EPCGLOBAL] EPCglobal, EPC Radio Frequency Identity Protocols Class 1
   1516 Generation 2 UHF RFID Protocol for Communications at 860 MHz-960
   MHz Version 1517 1.0.9, EPCglobal Standard, January 2005.

   [NIST-800-108] NIST Special Publication 800-108, Recommendation for
   Key Derivation Using Pseudorandom Functions.

   [SEC] S. Weis, S. Sarma, R. Rivest and D. Engels. "Security and
   privacy aspects of low-cost radio frequency identification systems"
   In D. Hutter, G. Muller, W. Stephan and M. Ullman, editors,
   International Conference on Security in Pervasive Computing - SPC
   2003, volume 2802 of Lecture Notes in computer Science, pages 454-
   469. Springer-Verlag, 2003.

   [HIP-TAG-EXP] Pascal Urien, Simon Elrharbi, Dorice Nyamy, Herve
   Chabanne, Thomas Icart, Francois Lecocq, Cyrille Pepin, Khalifa
   Toumi, Mathieu Bouet, Guy Pujolle, Patrice Krzanik, Jean-Ferdinand
   Susini, "HIP-Tags architecture implementation for the Internet of
   Things", AH-ICI 2009. First Asian Himalayas International Conference
   on Internet, 3-5 Nov. 2009.

   [BLIND] Dacheng Zhang, Miika Komu, "An Extension of HIP Base Exchange
   to Support Identity Privacy", draft-zhang-hip-privacy-protection-00,
   work in progress, March 2010.

9 Annex I

   This annex provides a sample code, for NFC RFIDs working at 13.56 Mhz
   and implementing a Java Virtual Machine.

9.1 Binary Interface with HIP RFIDs

   According to the ISO 7816 standards, embedded RFID applications are
   identified by an AID attribute (Application IDentifier) whose size
   ranges between 5 and 16 bytes.

   Commands exchanged between RFIDs and readers are named APDUs and are
   associated with a short prefix, whose size is usually 5 bytes
   referred as CLA, INS, P1, P2, P3.

   In our sample we choose an arbitrary value for the AID
   (11223344556601, in hexadecimal representation) and a unique command
   CLA=00, INS=C2, P1=00, P2=00. The P3 byte is set to null in order to
   trig the RFID (which resets its state machine and returns the I1
   packet, or a non null value when it pushes the R1 packet.

9.3 Exchanged data

   The reader selects the embedded HIP-RFID application.
   >> 00 A4 04 00 07 11 22 33 44 55 66 01
   << 90 00

   The reader trigs the first packet I1-T.

   >> 00 C2 00 00 00

   The RFID delivers the R1-T packet.

   << 3B 04 40 11 00 00 00 00 A3 12 9D 5E 28 16 67 4F FC 4F A8 08 4E 30
   55 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 90 00

   The reader forwards the R1-T packet to the HIP RFID.

   >> 00 C2 00 00 58 3B 0A 41 11 00 00 00 00 00 00 00 00 00 00 00 00 00
   00 00 00 00 00 00 00 A3 12 9D 5E 28 16 67 4F FC 4F A8 08 4E 30 55 E8
   04 00 00 20 00 06 68 46 95 15 02 10 32 C2 B7 8D 13 E7 53 F6 25 0F 09
   AD 7A BD 00 00 00 00 00 00 04 02 00 10 00 06 00 01 00 00 00 00 00 00
   00 00

   The RFID produces the I2-T packet.

   << 3B 13 40 11 00 00 00 00 A3 12 9D 5E 28 16 67 4F FC 4F A8 08 4E 30
   55 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 04 02 00 10 00
   06 00 01 00 00 00 00 00 00 00 00 04 00 00 20 00 06 71 3A DD 19 C4 CB
   59 D4 AF D0 2B FD F9 7C 2F 8A D1 23 32 E0 00 00 00 00 00 00 04 04 00
   20 00 06 70 DA C1 F7 0B CA 63 15 57 CB D7 AA 66 A9 FD 36 B4 1F DB E3
   00 00 00 00 00 00 04 06 00 20 00 06 A6 A7 00 67 5D FD A9 2F 3E 5C 00
   D6 B0 8A 55 A2 99 D8 86 79 00 00 00 00 00 00 90 00

9.3 Javacard code sample

```
package hiprfid;

// Author Pascal Urien

import   javacard.framework.*;
import   javacard.security.* ;


public class rfid extends Applet
{
 final static byte  SELECT          = (byte)0xA4 ;
 final static byte  INS-HIP         = (byte)0xC2 ;

 final static short R-T             = (short)0x400 ;
 final static short HIP-T-TRANSFORM = (short)0x402 ;
 final static short F-T             = (short)0x404 ;
 final static short Signature-T     = (short)0x406 ;
 final static short ESP-Transform   = (short)0x408 ;
 final static short ESP-Info        = (short)0x40A ;

 final static int ALIGN = 8;
 final static short len-r2 =(short)20;
 final byte[] algo1 = {(byte)0x00,(byte)0x01,(byte)0x00,(byte)0x00 };

 final byte[] ct1 =  {
 (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x01,
 (byte)'T',(byte)'y', (byte)'p',(byte)'e',
 (byte)' ',(byte)'0',(byte)'0',(byte)'0',(byte)'1',
 (byte)' ',(byte)'k',(byte)'e',(byte)'y' };

 final byte[] ct2 =  {
 (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x02,
 (byte)'T',(byte)'y',(byte)'p',(byte)'e',
 (byte)' ',(byte)'0',(byte)'0',(byte)'0',(byte)'1',
 (byte)' ',(byte)'k',(byte)'e',(byte)'y' };

 MessageDigest sha1=null ;
 RandomData rnd=null;
 byte[] DB =null;
 final static short DBSIZE=(short)200;
 final static short off-myHIT = (short)0    ;
 final static short off-rHIT =  (short)16   ;
 final static short off-R1   =  (short)32   ;
 final static short off-R2   =  (short)64   ;
 final static short off-kaut =  (short)96   ;
 final static short off-k    =  (short)128 ;
 final static short off-FT   =  (short)160 ;
```

```
 final byte[] HEADER= {
 (byte)0x3b,(byte)0x04,(byte)0x40,(byte)0x11,
 (byte)0x00,(byte)0x00,(byte)0x00,(byte)0x00 };


 final byte[] MyEPCCODE = {
 (byte)0x01,(byte)0x23,(byte)0x45,(byte)0x67,(byte)0x89,
 (byte)0xab,(byte)0xcd,(byte)0xef,(byte)0xcd,(byte)0xab };

 public void init(){
 try { sha1=MessageDigest.getInstance(MessageDigest.ALG-SHA,false);}
 catch (CryptoException e){sha1=null;}

 try { rnd = RandomData.getInstance(RandomData.ALG-SECURE-RANDOM);}
 catch (CryptoException e){}

 DB = JCSystem.makeTransientByteArray(DBSIZE,
                                      JCSystem.CLEAR-ON-DESELECT);

}

public short GetAttOffset(byte[] pkt, short off, short len,short att)
{ boolean more=true;
  short type=(short)0;
  short tl=(short)0;

  if (len <= (short)40) return (short)-1 ;

  while (more)
  { type = Util.getShort(pkt,off)              ;
    tl   = Util.getShort(pkt,(short)(off+2));
    if (type == att) return off     ;
    off =(short)(off+tl) ;
    if (off >= (short)(off+len))more=false;
  }

 return -1;
}


public static short GetPadLength(short size)
{
 if ( (short)(size % ALIGN) == (short)0) return (short)0;
 return (short)(ALIGN - size % ALIGN );
 }


public static short Set_Att(short att, byte[] ref-att, short off-att,
                         short len-att, byte[] pkt, short off)
{
  short tl = (short) (len-att + 6) ;
```

```
   short tp = GetPadLength(tl)        ;

   tl= (short) (tp+tl);

   Util.setShort(pkt,off,att)   ;
   Util.setShort(pkt,(short)(off+2),tl);
   Util.setShort(pkt,(short)(off+4),tp);

   if (ref_att != null)
   Util.arrayCopy(ref-att,off-att,pkt,(short)(off+6),len-att);
   else
   Util.arrayFillNonAtomic(pkt,(short)(off+6),len-att,(byte)0);

   if (tp != (short)0)
   Util.arrayFillNonAtomic(pkt,(short)(off+6+len-att),tp,(byte)0);

   return tl ;
  }


public void process(APDU apdu) throws ISOException
{
 short len=(short)0, readCount=(short)0;
 short off=(short)0,pad=(short)0,len-r1=(short)0;
 short size=(short)0;

 byte[] buffer = apdu.getBuffer() ; // CLA INS P1 P2 P3

 byte cla = buffer[ISO7816.OFFSET_CLA];
 byte ins = buffer[ISO7816.OFFSET_INS];
 byte P1  = buffer[ISO7816.OFFSET_P1] ;
 byte P2  = buffer[ISO7816.OFFSET_P2] ;
 byte P3  = buffer[ISO7816.OFFSET_LC] ;

 switch (ins)
 {
  case SELECT:
  size = apdu.setIncomingAndReceive();
  return;

 case INS_HIP:

  if (P3 == (byte)0)
  {
   rnd.generateData(DB,off_myHIT,(short)16);
   Util.arrayCopy(HEADER,(short)0,buffer,(short)0,(short)8);
   Util.arrayCopy(DB,off-myHIT,buffer,(short)8,(short)16)  ;
   Util.arrayFillNonAtomic(DB,(short)24,(short)16,(byte)0) ;
   apdu.setOutgoingAndSend((short)0,(short)40)             ;
   break;
  }
```

```
            else
            {
            size = apdu.setIncomingAndReceive();
            len  = Util.makeShort((byte)0,buffer[6]);
            len  = (short)(len << 3);
            len  = (short)(len+(short)8)   ;

            if (len != size) ISOException.throwIt(ISO7816.SW-DATA-INVALID) ;
            size = (short)(len-(short)40);

            // HEADER 00...08
            // HIT-S  08...24
            // HIT-D  24...40

            Util.arrayCopy(buffer,(short)13,DB,off_rHIT,(short)16);
            off= GetAttOffset(buffer,(short)45,size,R-T);
            if (off==(short)-1) ISOException.throwIt(ISO7816.SW-DATA-INVALID) ;
            len = Util.getShort(buffer,(short)(off+2));
            pad = Util.getShort(buffer,(short)(off+4));
            len = (short)(len-pad-6);

            len-r1=len;
            Util.arrayCopy(buffer,(short)(off+6),DB,off-R1,len);
            off= GetAttOffset(buffer,(short)45,size,HIP-T-TRANSFORM)        ;

            if (off==(short)-1) ISOException.throwIt(ISO7816.SW-DATA-INVALID) ;
            len = Util.getShort(buffer,(short)(off+2));
            pad = Util.getShort(buffer,(short)(off+4));
            len = (short)(len-pad-6);

            // algo=Util.getShort(buffer,(short)(off+6)
            rnd.generateData(DB,(short)(off-R1+len-r1),len-r2); // r1 || r2

            Util.arrayCopy(MyEPCCODE,(short)0,buffer,
                                    (short)0,(short)MyEPCCODE.length);

            hmac(DB,off_R1,(short)(len-r1 + len-r2),
                 buffer,(short)0,(short)MyEPCCODE.length,
                 sha1,
                 DB,off-k);

            Util.arrayCopy(ct1,(short)0,buffer,(short)0,(short)ct1.length);

            hmac(DB,off_k,(short)20,
                 buffer,(short)0,(short)ct1.length,
                 sha1,
                 DB, off-FT);

            Util.arrayCopy(ct2,(short)0,buffer,(short)0,(short)ct2.length);
```

```
  hmac(DB,off-k,(short)20,
       buffer,(short)0,(short)ct2.length,
       sha1,
       DB, off-kaut);

  Util.arrayCopy(HEADER,(short)0,buffer,
                 (short)0,(short)HEADER.length);

  Util.arrayCopy(DB,off-myHIT, buffer, (short)8,(short)16);
  Util.arrayCopy(DB, off-rHIT, buffer,(short)24,(short)16);

  off=(short)40;
  len = Set-Att(HIP-T-TRANSFORM,algo1,
                (short)0,(short)algo1.length,buffer,off);
  off = (short)(off+len);
  len = Set-Att(R-T,DB,(short)(off-R1+len-r1),len-r2,buffer,off);
  off = (short)(off+len);
  len = Set-Att(F-T,DB,off-FT,(short)20,buffer,off);
  off = (short)(off+len);
  len = Set-Att(Signature-T,null,(short)0,(short)20,buffer,off);
  size= (short)(off+len);
  buffer[1] = (byte) (size >>3);

 hmac(DB,off-kaut,(short)20,
      buffer,(short)0,size,
      sha1,
      buffer,(short)(off+6));

apdu.setOutgoingAndSend((short)0,size);
  break;
}

 default:
 ISOException.throwIt(ISO7816.SW-INS-NOT-SUPPORTED);
  }

}

protected rfid(byte[] bArray,short bOffset,byte bLength)
{init();
 register();
}

public static void install( byte[] bArray, short bOffset, byte
bLength )
{
new rfid(bArray,bOffset,bLength);
}
```

```
    public boolean select()
    {
    return true;
    }

    public void deselect()
    {
    }
```

Author's Addresses

    Pascal Urien
    Telecom ParisTech
    23 avenue d'italie, 75013 Paris, France

    Email: Pascal.Urien@telecom-paristech.fr

    Gyu Myoung Lee
    Telecom SudParis
    9 rue Charles Fourier, 91011 Evry, France

    Email: gm.lee@it-sudparis.eu

    Guy Pujolle
    Laboratoire d'informatique de Paris 6 (LIP6)
    4 place Jussieu
    75005 Paris France

    Email: Guy.Pujolle@lip6.fr