

METHOD AND DEVICE TO TRANSFER A VIDEO STREAM BETWEEN A HOST DEVICE AND AN ELECTRONIC DESCRAMBLING DEVICE

Introduction

Multimedia contents may be transmitted in various ways from a provider to an end user. In order to
5 protect this valuable content, scrambling (encrypting) may be performed to the content so that only
authorized persons can have access to it.

Reception of the scrambled multimedia content may be done on different devices such as a personal
computer, a tablet, a smartphone, or other devices that may not have the necessary descrambling
(decrypting) capabilities to access the content.

10 Brief description of the invention

In one embodiment, an electronic descrambling device may be connected to a host device and
configured to descramble multimedia content. An electronic descrambling device is a portable
module comprising electronic chips connected via a communication port with the host device. It is
generally called "dongle". The electronic chip comprises a descrambling engine and memories to
15 temporary store the incoming data chunk. According to a particular embodiment, the electronic
descrambling device is a USB dongle and the video stream is a MPEG stream.

Currently, transfer protocols to an electronic descrambling device are not adapted to the continuous
exchange of data at a bit rate compatible with video streams. They have been designed to
accommodate the transfer of storage data. To address this problem, it is proposed an electronic
20 descrambling device connected to a host device to implement a method to descramble at least one
video stream originating from at least one producer, having a producer ID, said host device
comprising a controller comprising a bulk pipe out and a bulk pipe in transfer, comprising, at the host
device :

- for each producer, defining a chunk of data from the video stream,

- forming a chain of out transfer, the chain of out transfer comprising a chain out header (oqH) linked
25 with a plurality of chain out descriptors (oqTD), the first chain out descriptor pointing to a out
description packet containing the at least one producer ID, the second and subsequent chain out
descriptor pointing to the chunk of data, each chunk of data being of a particular producer,

- forming, by the host device, a chain of in transfer, the chain of in transfer comprising a chain in
30 header (iqH) linked with a plurality of chain in descriptors (iqTD), each chain in descriptor pointing to
a descrambled chunk of data, and the last chain in descriptor being configured to generate an
interrupt,

- transferring the chain of out transfer by the controller to the electronic descrambling device, thus transferring to the electronic descrambling device the scrambled data chunk,

at the electronic descrambling device :

- receiving the description packet by the electronic descrambling device and identifying by the

5 electronic descrambling device key data associated with the at least one chunk,

- using the key data to descramble the scrambled data chunk,

- returning the descrambled data to the host device,

at the host device :

-receiving the chain of in transfer by the controller, thus waiting from the electronic descrambling
10 device the chunk descrambled data,

- storing the received descrambled data chunk at an address indicated by the correspond chain in descriptor (iqTD),

- triggering an interrupt by the reception of the last descrambled chunk.

The format of the multimedia content is preferably an MPEG stream. An MPEG stream typically
15 comprises several sub-streams having each a different packet identifier (PID). Examples of sub-streams are audio, video, data, and caption. Sub-streams pertain to the same multimedia content from a channel. A MPEG stream can comprise several channels, the description of each sub-stream being located into the Program Map Table (PMT). As far as the scrambling of data is concerned, the keys loaded into the electronic descrambling device are dedicated to a sub-stream and identified
20 using the producer ID and the PID.

Brief Description of the Figures

The following detailed description will be better understood thanks to the attached figures in which:

- figure 1 illustrates a block diagram of the host device in connection with the electronic descrambling device,

25 - figure 2 illustrates the format of the chain in and out queue according one example of the invention,

- figure 3a illustrates a first embodiment of the transfer chain,

- figure 3b illustrates a second embodiment of the transfer chain,

- figure 4 illustrates the management of the queues in a buffer,

30 - figure 5 illustrates an example of the modules in a electronic descrambling device.

Detailed description

Figure 1 illustrates the different components of one embodiment of the invention. The Host (which may be for example a home multimedia receiver) comprises various layers. The top layer may be the TV Native Application. A Middleware layer may include a DVB Middleware and the Media Player modules. The OS Drivers may be lower layers containing the Stream Management Unit in charge of executing one embodiment of the invention. The communication layer SoC may be configured for the handling of the MPEG streams and may include the communication module (USB controller) with the electronic descrambling device. In the following description, the electronic descrambling device will be named “dongle”.

Figure 2 describes one example of the formatting of a chain for a controller, in particular a USB controller. The application layer, in the host device, receives the MPEG stream (or a plurality of MPEG streams) and formats the chain in and chain out (CT) in order to instruct the USB controller. The chain out starts with a chain out header (oqH) playing the role of the link with the chain out descriptors (oqTD1 to oqTDn). The first chain out descriptor (oqTD1) is pointing to the out description packet containing the at least one producer ID. The application layer is in charge of composing the chains and extracts the producer ID from the MPEG stream. The application layer then loads the producer ID into the out description packet (oDP).

The first chain out descriptor (oqTD1) points to the second chain out descriptor (oqTD2). This second chain out descriptor (and the following chain out descriptors) points to the chunk of data extracted from the MPEG stream. The application layer handling the MPEG stream splits the stream into chunks of data and loads them into the buffer memory at an address loaded into the chain out descriptor.

The application layer can load more than one chunk into the chain in accordance with the maximum number of chain out descriptors handled by the chain.

According to the example of Figure 3a and 3b, when the application layer is processing more than one MPEG stream, the producer ID of each stream is loaded into the out descriptor packet (oDP). This is illustrated by the figure 3b in which three streams are processed and the producer ID1, the producer ID2 and the producer ID3 are loaded into the out descriptor packet.

The MPTS1 Data, forming the first data chunk, is then pointed by the second chain out descriptor (oqTD2), the MPTS2 Data, forming the second data chunk, is then pointed by the third chain out descriptor (oqTD) and the MPTS3 Data, forming the third data chunk, is then pointed by the fourth chain out descriptor (oqTD1).

The application layer also configures the chain in. The chain in and chain out are connected together and form a single event. Once configured, the application layer calls the USB controller to execute the chain operations. The chain in is a chain starting with a chain in header (iqH) linked with a plurality of chain in descriptors (iqTD). Each chain in descriptor points to a chunk of data resulting from the processing (i.e. descrambling) by the USB dongle.

The last chain in descriptor (iqTDn) is configured to trigger an interrupt when the USB controller receives the last chunk of processed data by the USB dongle.

Each chain in descriptor is associated with a MPEG stream. Depending on the implementation of the USB controller, the size of data chunk can vary. According to one example, a data chunk comprises a plurality of packets. Each packet represents a USB transaction. According to one example of forming a chain of packets, each packet has a size equal to the nominal size (e.g. 512 Bytes), except the last packet. The non-nominal size of the packet indicates that this is the last packet of the data chunk. This is illustrated in figure 2 by the box DCo1 to DCop. The last packet of the data chunk has a size different than the nominal size, thus indicating that it is the last packet of the data chunk.

According to another example of realization, the chain out transfer is configured such as the producer ID is placed in the data chunk, into the first packet. The producer ID is placed in a data header preceding the data chunk and both are concatenated by the USB controller and transferred in a single USB transaction.

This is the case illustrated in figure 3A. The first chain out descriptor points to at least two packets, one being the packet containing the producer ID and the second one (and further ones) being the packets of the data chunk of the MPEG stream. The chain in is the same as in the first example.

When the USB controller is supporting both 3a and 3b, 3a and 3b share a common field in their respective header allowing the dongle USB controller to differentiate between a 3a and a 3b chain of descriptor. This allows a host to choose the most efficient transfer for a specific processing.

Once the chain out and in is configured, the USB controller receives a signal from the application layer and the chain is processed. The transfer to the USB dongle is executed by the USB controller for example using the bulk in and out transfer. The USB controller activates and transfers the data as instructed.

The USB dongle receives the chain out and retrieves the producer ID from the corresponding packet. With the producer ID, the USB dongle can retrieve the key data corresponding to the data chunk. The USB dongle receives through another USB channel, key data related to the MPEG streams to proceed.

The USB dongle comprises a key table populated with the key data for each MPEG stream. According to a first example, the key table comprises for each producer ID, the key data related to this producer. Preferably, the key data comprises an odd and an even key, one being currently used and the other one ready for the next key change. A bit in the chunk data indicates the current key to be used.

This key table can further contain a packet identifier PID (or program ID) in case one producer ID is associated with a plurality of key data. The table will be then as follows :

Producer ID	PID ID	Key odd	Key even
PRID1	PID12	Value	Value
PRID1	PID15	Value	Value
PRID2	PID3	Value	Value
PRID3	PID10	Value	Value

An example of the value of the keys is a 128 bits pseudorandom number.

- 10 In the above example, the producer PRID1 is associated with an MPEG stream comprising two sub-streams needing key data to be descrambled. Each sub-stream is identified by a PID identifier or program ID (in our example PID12 and PID15), each sub-stream being associated with different key data.

- 15 The key table is received by the host device from an authorization server or can be extracted from one or more MPEG stream. According to an embodiment of the invention, the USB dongle comprises the key(s) necessary to decrypt the encrypted key data. The encrypted key data can be in the form of ECM (Entitlement Control Message) extracted from the MPEG stream by the host and transferred to the USB dongle.

- 20 According to another embodiment, the host can request a license file from an authorization server and can pass the license file to the USB dongle once received from the authorization server. The license file is decrypted and the key data are used to populate the key table. The USB dongle can comprise a personal key to decrypt the encrypted key data (license or ECM).

- 25 Once the USB dongle has descrambled the data chunk using the corresponding key data, the descrambled data chunk are sent to the host device, in particular to the USB controller. The latter stores the received data at the address indicated in the first chain out descriptor. As a consequence, a data chunk DCo1 is transferred by the USB controller to the USB dongle and when the USB dongle

returns the data chunk DCi1, the USB controller stores it at the address indicated by the first chain in descriptor iqTD1.

When the last data chunk has been stored in the chain in queue the USB controller initiates an interrupt to inform the application layer.

- 5 The application layer can then retrieve the descrambled data chunk and can deliver it to a consumer such as a multimedia player.

It is to be noted that a MPEG stream is loaded into a stream buffer as illustrated at the figure 4. The host can process several MPEG streams in parallel and the figure 4 illustrates two MPEG streams from two producers (Producer#1, Producer#2). The application layer is in charge of the management of the Stream Buffer#1 and #2. A buffer is dedicated to one producer and the application layer
10 mainly deals with the pointers of the buffer. The buffer is filled with data coming from an MPEG source. When the chain in and chain out are defined, the USB controller process the data chunk (chunk#1 for the Producer#1 and chunk#2 for the Producer#2) so that more than one MPEG source can be processed in parallel. The consumer is using the buffer architecture and handles its own
15 pointers to the descrambled data.

According to the illustrated example, the stream buffer is the same for the scrambled and descrambled data. However, in accordance with another example, the buffer for the scrambled data is independent to the buffer of the descrambled data. In this case, the USB controller, while receiving the descrambled chunk from the USB dongle, stores them into the in buffer.

- 20 This formatting and handling of chain buffer is particularly adapted to USB 2.0 transmission. However, other protocol layer can be used to transfer the data based on stream buffer and chain pointer and descriptor as described above.

Figure 5 illustrates a block diagram of the dongle. The dongle comprises four main blocks, the I/O interface block, the processing block PRO_M, the memory MEM and the descrambling module
25 DES_M.

The I/O interface is connectable with the host device to receive the data via the dedicated USB signals. The processing block PRO_M is in charge of directing the flow of data entering and exiting via the I/O Interface to the memory MEM. The memory acts as a buffer memory and the pointers (in and out pointers) are handled by the processing block. The data chunks are stored in the memory and the
30 pointer for this chunk is passed to the descrambling module. The processing block also keep track of the producer ID for said chunk of data and loads the related keys into the descrambling module. The descrambling module, when the chunk is descrambled, stores the descrambled chunk and sends a signal to the processing module. The pointer is then passed to the I/O interface which routes the

descrambled chunk into the I/O Interface. The processing block is in charge of executing the chain's instruction. The structure of the chain is known by the processing block and the elements of the chain extracted so that the processing block knows for each data chunk, which producer ID is related to. For that purpose, the processing block is connected with a key memory K_MEM to store the keys received from the host device via the USB Interface.

According to another embodiment, the keys memory K_MEM is directly connected to the descrambling module and this module, when the processing block instructs to descramble a data chunk identified by producer ID, loads the keys related to said producer ID in the key registers.

Although an overview of the inventive subject matter has been described with reference to specific example embodiments, various modifications and changes may be made to these embodiments without departing from the broader spirit and scope of embodiments of the present invention. For example, various embodiments or features thereof may be mixed and matched or made optional by a person of ordinary skill in the art. Such embodiments of the inventive subject matter may be referred to herein, individually or collectively, by the term "invention" merely for convenience and without intending to voluntarily limit the scope of this application to any single invention or inventive concept if more than one is, in fact, disclosed.

The embodiments illustrated herein are believed to be described in sufficient detail to enable those skilled in the art to practice the teachings disclosed. Other embodiments may be used and derived therefrom, such that structural and logical substitutions and changes may be made without departing from the scope of this disclosure. The Detailed Description, therefore, is not to be taken in a limiting sense, and the scope of various embodiments is defined only by the appended claims, along with the full range of equivalents to which such claims are entitled.

Claims

1. An electronic descrambling device for descrambling a video stream, the video stream originating from a producer, said electronic descrambling device comprising:

processing circuitry configured to

receive a chain out transfer from an I/O interface connectable with a host device, and thus receive a scrambled data chunk, wherein the scrambled data chunk is of a particular producer and defined from the video stream, and the chain out transfer comprises a chain out header linked with a plurality of chain out descriptors, a first chain out descriptor pointing to an out description packet containing a producer ID, second and subsequent chain out descriptors pointing to a corresponding data chunk,

receive the out description packet;

retrieve the producer ID from said out description packet;

identify key data associated with the scrambled data chunk using the retrieved producer ID;

use the identified key data to descramble the scrambled data chunk, and

return the descrambled data to the I/O interface connectable with the host device.

2. The electronic descrambling device according to claim 1, wherein the electronic descrambling device is a USB Dongle.

3. The electronic descrambling device according to claim 1, further comprising a memory storing a key table populated with, for each producer ID of a plurality of producer IDs, key data related to said producer ID.

4. The electronic descrambling device according to claim 3, wherein the key table contains a packet identifier PID or a program ID, in case one producer ID is associated with a plurality of key data.

5. The electronic descrambling device according to claim 1, further comprising a memory storing a personal key, wherein the processing circuitry is further configured to receive the key data encrypted from the host device, and decrypt the encrypted key data using the stored personal key.

6. The electronic descrambling device according to claim 5, wherein the processing circuitry is further configured to obtain the encrypted key data in a form of an Entitlement Control Message.

7. The electronic descrambling device according to claim 5, wherein the processing circuitry is further configured to obtain the encrypted key data in a license file.

8. The electronic descrambling device according to claim 1, further comprising the I/O interface connectable with the host device, a memory acting as a buffer memory and configured to store data chunks,

wherein the processing circuitry is further configured to direct a flow of data entering or exiting via the I/O interface to the memory, and

the processing circuitry includes descrambling circuitry, wherein the processing circuitry is further configured to handle in and out pointers of the buffer memory, pass a pointer for a data chunk stored in the memory to the descrambling circuitry, keep track of the producer ID for said data chunk, and load the key data into the descrambling circuitry.

9. The electronic descrambling device according to claim 1, further comprising a key memory to store the key data received from the host device.