(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 October 2002 (17.10.2002)

PCT

(10) International Publication Number
WO 02/082261 A2

(51) International Patent Classification⁷: G06F 9/40

(21) International Application Number: PCT/IB02/01061

(22) International Filing Date: 4 April 2002 (04.04.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
01/04643 5 April 2001 (05.04.2001) FR
02/02314 18 February 2002 (18.02.2002) FR

(71) Applicant *(for all designated States except US)*: **SCHLUMBERGER SYSTÈMES** [FR/FR]; 50, avenue Jean Jaurès, F-92120 Montrouge (FR).

(71) Applicant *(for MC only)*: **SCHLUMBERGER MALCO, INC.** [US/US]; 9800 Reistertown, Owing Mills, MD 21117 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **CABOS, Philippe** [FR/FR]; 44 rue Périer, F-92120 Montrouge (FR). **DEROUET, Odile** [FR/FR]; 76 rue des Carves, F-92120 Montrouge (FR).

(74) Common Representative: **SCHLUMBERGER SYSTÈMES**; C/O GUILLERM, Patrick, 50, avenue Jean Jaurès, F-92120 Montrouge (FR).

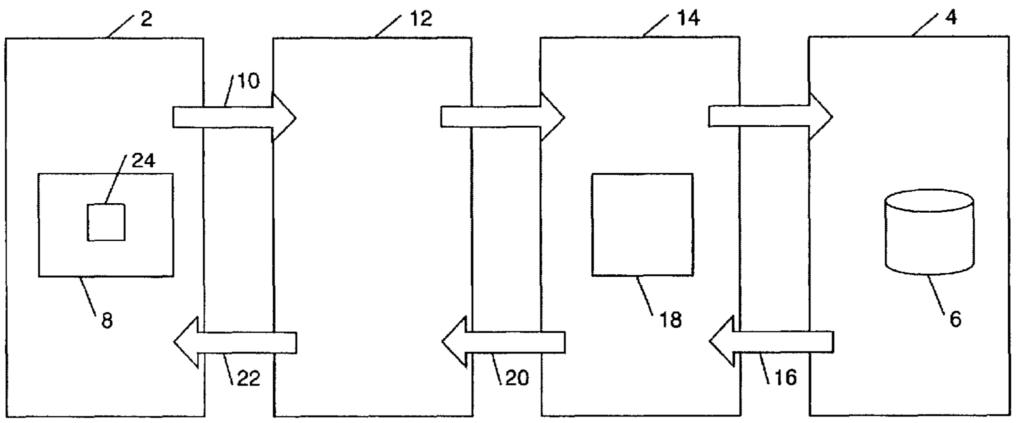
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:
— *of inventorship (Rule 4.17(iv)) for US only*

Published:
— *without international search report and to be republished upon receipt of that report*

(54) Title: METHOD AND DEVICE FOR COMPRESSION AND DECOMPRESSION OF CODES OF AN APPLICATION, WRITTEN IN HIGH LANGUAGE, ESPECIALLY IN MOBILE TELEPHONY.



(57) **Abstract:** A method of loading an application comprising codes onto a subscriber identification module (8), the method comprising: - a compression step in which each code of a pre-defined set of codes is replaced by a reference to that code so as to obtain a compressed application; - a loading step in which the compressed application is loaded onto the subscriber identification module; - a decompression step in which each reference comprised in the compressed application is replaced by the code that is referred to so as to reconstitute the application; - and a storage step in which the application is stored in a memory of the subscriber identity module.

WO 02/082261 A2

METHOD AND DEVICE FOR COMPRESSION AND DECOMPRESSION OF CODES OF AN APPLICATION, WRITTEN IN HIGH LEVEL LANGUAGE, ESPECIALLY IN MOBILE TELEPHONY

This invention concerns the compression and decompression of codes of an application, such as an application written in high level language, especially in mobile telephony.

It has a general application in data processing and more especially in mobile telephony.

Most mobile telephony operators require an increasing number of services using the SMS (Short Message Service) channel of their network to transmit application data written in high level language (for example applets written in JAVA language) between the subscriber's mobile device (telecommunication terminal generally equipped with a subscriber identification module) and a computer equipment managing this service.

The mobile telephony communication standard, for example GSM (Global System for Mobile communication) allocates a very small passband (only several hundred bytes per second) for the SMS channel which was initially only planned for transmission of text messages not exceeding 180 bytes.

The operator's equipment is quickly congested by a non negligible quantity of data to be transmitted on this low speed SMS channel. Consequently, there is a high latency time for the subscriber.

Improving the performance involves extremely costly investments in network equipment which the operator finds more and more difficult to amortise. In addition, it is always limited by the maximum speed of the SMS channel.

The Applicant therefore decided to study the problem of reducing the number of data bytes transmitted, especially on this SMS channel.

This invention provides a solution to this problem.

It concerns a data processing method for a mobile communication device equipped with a subscriber identification module, the said data including codes belonging to an application written in high level language from a remote transmitting set.

The subject of the invention is a method of loading an application comprising codes onto a subscriber identification module (8), the method comprising:

- a compression step in which each code of a pre-defined set of codes is replaced by a reference to that code so as to obtain a compressed application;
- a loading step in which the compressed application is loaded onto the subscriber identification module;
- a decompression step in which each reference comprised in the compressed application is replaced by the code that is referred to so as to reconstitute the application;
- and a storage step in which the application is stored in a memory of the subscriber identity module.

In our illustrated example, the method used is of the type which includes the following known steps:

- equip the subscriber identification module with an interpreter able to interpret the codes of an application written in high level language;

- equip the subscriber identification module and the transmitting set with a communication protocol according to which the mobile device is able to obtain the codes of a desired application from the said transmitting set.

According to a first embodiment, the method also includes for the transmitting set the following steps:

- process dynamically at least some byte sequences contained in the said codes so as to recognise in these byte sequences, the byte sequences identically duplicated at least twice, and

- for at least one duplicated sequence, define a compression sequence, and store this compression sequence at an address chosen in a compression table, replace at least one duplicated sequence in the original data by the address (the reference) of the corresponding compression sequence, and send the codes so compressed to the mobile subscriber device.

In our first embodiment, the method includes, for the subscriber identification module, the following steps:

-1) receive the codes so compressed from the transmitting set and reassemble them;

-2) recognise in the data so reassembled the non duplicated byte sequences and the compression sequences; and

-3) store each non duplicated byte sequence in a decompression table at a respective address, and replace at least one compression sequence by the corresponding non duplicated byte sequence located in the decompression table at the address indicated in the compression sequence, and thereby obtain decompressed codes.

According to a second embodiment, the method includes for the transmitting set the following step:

- process statically at least some byte sequences of the said codes, and convert the said byte sequences into bit strings (the reference) according to a predetermined conversion rule, replace each byte sequence by its corresponding bit string, and transmit the codes so converted to the mobile subscriber device.

Advantageously, the dynamic and the static process are implemented more or less simultaneously, the codes transmitted to the mobile subscriber device being both converted and so compressed.

In our second embodiment, the bit string length depends on the frequency of the codes in the application to be processed.

Advantageously, in this second embodiment, the method includes for the subscriber identification module the following step:

-receive the codes so converted from the transmitting set, reassemble them, and replace at least one bit string by the corresponding byte sequence according to a predetermined conversion function.

Advantageously, this last step and the above step 3 are implemented simultaneously, the codes processed in the subscriber identification module being converted and decompressed more or less simultaneously.

The data can be of different types. For example, when the data is applets, it is planned according to the invention to equip the transmitting set with an applet server, and to download applets from the said server to the subscriber identification module.

In another example, in which the data is messages, the method according to the invention also includes a first additional step planning a gateway connected to a message management centre and positioned between the mobile device and a remote application server, and a second additional step planning the use by the mobile subscriber device, the management centre, the gateway and the application server of a communication protocol according to which the mobile device is able to obtain the codes of a desired application following a request to the remote server via the said message management centre and the said gateway, the gateway compressing the messages at least according to steps c) and d) of the above-mentioned method.

This invention also concerns a data processing device for loading an application comprising codes onto a subscriber identification module, the method comprising:

- Means for replacing each code of a pre-defined set of codes by a reference to that code so as to obtain a compressed application;
- Means for loading the compressed application onto the subscriber identification module;
- Means for replacing each reference comprised in the compressed application by the code that is referred to so as to reconstitute the application; and means for storing the application in a memory of the subscriber identity module.

The invention also concerns a subscriber identification module being arranged to effect

- the above decompression step in which each reference comprised in the compressed application is replaced by the code that is referred to so as to reconstitute the application;
- and the above storage step in which the application is stored in a memory of the subscriber identity module.

The invention also concerns a server being arranged to effect

- the above compression step in which each code of a pre-defined set of codes is replaced by a reference to that code so as to obtain a compressed application;
- the above loading step in which the compressed application is loaded onto the subscriber identification module;

The invention also concerns a computer program product for a data processing device, the computer program product including an instruction set which when the instruction set is loaded in the data processing device, makes the data processing device perform the above steps of

- decompression in which each reference comprised in the compressed application is replaced by the code that is referred to so as to reconstitute the application;
- storage in which the application is stored in a memory of the subscriber identity module

The invention also concerns a computer program product for a data processing device, the computer program product including an instruction set which when the instruction set is loaded in the data processing device, makes the data processing device perform the the above steps of

- -compression in which each code of a pre-defined set of codes is replaced by a reference to that code so as to obtain a compressed application;
- loading in which the compressed application is loaded onto the subscriber identification module;

Other features and advantages of the invention will appear on reading the detailed description below and the drawings in which:

- figure 1 is a diagrammatic representation of the architecture of the processing device according to the invention;

- figure 2 is a flowchart illustrating the dynamic compression according to the invention;

- figure 3 is a flowchart illustrating the static compression according to the invention;

- figure 4 is a flowchart illustrating the dynamic decompression according to the invention; and

- figure 5 illustrates the static decompression according to the invention.

- figure 6 illustrates another embodiment of the invention.

Figure 1 represents an SMS message processing device according to the invention finding an application for example in the second generation GSM mobile telephony network.

In this network, a mobile subscriber terminal 2 can access services or applications 6 (applets) stored on a remote application server 4.

For example, the applications 6 are pages written in WML (Wireless Markup Language) which is a content description language for small portable wireless devices. As a variant, the applications are written in SATML (Simalliance Toolkit Markup Language) produced by the company Simalliance. This company, founded in 2000, writes technical specifications used to share the advantages of the Internet world and the world of mobile telephones and more especially the subscriber identification modules (SIM).

The most important technical specifications can be obtained from http://www.simalliance.org/.

The subscriber terminal 2 can access pages of the Internet from a simple mobile communication device 2 which does not have a WAP (Wireless Application Protocol) type browser. Unlike this type of mobile device where the page browser is integrated in the telephone terminal, the terminal 2 of the invention includes a SIM TOOLKIT type browser 8 as described in the GSM specification 11.14.

When a subscriber wants to obtain a service or an application 6 such as a page of an Internet site, the terminal 2 transmits a request 10 in SMS format as described in the GSM specifications 11.14, 03.48 and 03.40 and S@T (Simalliance toolbox specifications) 01.22, 01.20. This request 10 is transmitted via the SMS message management centre 12 of the GSM network to a gateway 14 which handles the transmission of this type of request 10 to the remote server 4 located on the Internet which hosts the requested page. The content of the page 16 is then sent to the gateway 14 which encodes it and transfers the interpretable code 20 so generated to the SMS message management centre 12 of the subscriber's telephony operator. The management centre 12 then transmits the encoded page 22 to the terminal 2, whose interpreter 24 is able to interpret the interpretable code so received.

The Applicant has observed that to reduce the user's waiting time, the size of the data transmitted by the server 12 on the SMS channel of the GSM network to mobile communication terminals 2 must be reduced.

To reach this objective, the stream of data produced by encoding the page at the gateway 14 is compressed and decompressed in a complementary way in the subscriber module of the mobile device.

In reference to figure 2, the flowchart of the compression algorithm in the gateway according to the invention includes the following steps.

In step E1, the codes of the application 16 (bytecode) from the application server 4 are read and at least some byte sequences contained in the said codes are processed dynamically. Preferably, all bytes are processed.

In step E2, the byte sequences identically duplicated at least twice in these byte sequences are recognised.

In step E3, for at least one duplicated sequence, a compression sequence is defined, and this compression sequence is stored at an address chosen in a compression table.

In step E4, at least one duplicated sequence in the original message is replaced by the address of the corresponding compression sequence, and the codes so compressed are sent to the mobile device.

In other words, the redundant bit patterns are detected dynamically. In our example, a table or dictionary is built as the file is being read. The algorithm learns the bit patterns of the file during the read. When a character string already read is detected, the address of this string is encoded in a table, instead of the string itself.

In our example, the procedure is as follows: start to read the text and store it in a buffer memory. As each new character is read, check whether it is already in the table. If it is not, add it to the table. If it is, write its address in the compressed file together with a tag indicating that it is an address and check whether the following characters coincide.

For example, the table has a capacity of about 512 bytes. The table is written in RAM. As soon as the table exceeds this limit, go back to the start of the table.

The compression according to the invention can be implemented on a text written in hexadecimal with for example the following content "ABCDEF12345678ABCDEF1234".

At step 0, the buffer memory is initialised to 000...0.

At step 1, since byte AB is not in the buffer memory, the buffer memory is equal to AB000...0.

At step 2, since byte CD is not in the buffer memory, the buffer memory now contains ABCD000...0, and so on up to step 8 where, since byte 78 is not in the buffer memory, the buffer memory contains ABCDEF12345678000...0.

At step 15: ABCDEF1234 is in the buffer memory. Now write the text previously read with an index on the number of bytes written together with a tag indicating the number of redundant bytes and their address.

The file so compressed becomes 07ABCDEF123456788A00. The file includes a start tag BA1 and an end tag BA2.

The start tag BA1 is a tag which locates the non compressed text and its size. For example, tag BA1 is encoded according to a format (one byte) in which the first bit is 0 to indicate that the next text is not compressed, and bits 2 to 8 indicate the size of the next text. In the example, tag BA1 is "07" therefore indicating that the next seven bytes are not compressed.

The end tag BA2 is a tag which indicates the compression sequence and its address in the table. This tag BA2 is encoded for example of two bytes. The first bit set to 1 indicates that it is an address. Bits two to seven of the first byte indicate the number of consecutive bytes and bits eight to sixteen represent the address in the buffer memory. For example, tag BA2 equal to "8A00" indicates that there are five redundant bytes at address 0.

In reference to figure 3, the compression algorithm according to the invention also includes a static processing loop.

In this loop, at least some byte sequences of the codes are processed statically (step E10).

Then (step E11), the said byte sequences are converted into bit strings according to a predetermined conversion rule.

In step E12, each byte sequence is replaced by its corresponding bit string and the codes so converted are transmitted to the mobile device.

This static algorithm is simpler than the dynamic algorithm described in reference to figure 2.

To improve efficiency, static compression is carried out after dynamic compression. In fact, dynamic compression does not compress, or compresses very little, the tags used in the files processed, such as the DECK files described in the Simalliance specifications. These tags are common to all files and therefore appear more frequently than the others.

A statistical evaluation of the frequency of appearance of these tags can be calculated beforehand. A table is then created where each character is encoded on a number of bits inversely proportional to its frequency.

Note that the interpretable codes (bytecode), in this case tags, characters or other are generally encoded on one byte. The result of converting the byte sequences of the bytecodes into bit strings of size less than 8 bits is to reconvert all the bytecodes.

For example, the byte FF can be represented by the bit string 11. In order to differentiate this string from the others during the decoding, none of the remaining characters can start with this string. Consequently, characters starting with 11 must be encoded again. Weights must therefore be allocated to each remaining character.

Once the part of the file or text including tags has been processed, processing can start on the pages composed of text written in a given language. However, unlike the tagged part of the files, it is impossible to know a priori the language used in the text.

In our example, punctuation characters are placed at the start of the file since many languages share the same punctuation. We then consider that English is the language most commonly used and therefore encode the characters according to their frequency of appearance in this language. It is also possible to store the static table in the EEPROM memory of the subscriber identification module so that each operator can calculate this table according to the user's country and language.

In reference to figure 4, the decompression algorithm is symmetric to the compression.

In step E20, the codes so compressed are received from the gateway 14, via the management centre 12, and are reassembled since they do not necessarily arrive in the correct order. In other words, the compressed file is fully written in EEPROM memory and then decompressed.

In step E21, the non duplicated byte sequences and the compression sequences are recognised in the message so reassembled.

In step E22, each non duplicated byte sequence is stored in a decompression table at a respective address, and at least one compression sequence is replaced by the corresponding non duplicated byte sequence located in the decompression table at the address indicated in the compression sequence, thereby obtaining decompressed codes.

In our example, the decompression table is built in RAM. A tag indicates the type of byte sequence, i.e. address (compression sequence) or character (non duplicated sequence). If it is an address, the corresponding value is read in the decompression table and processing continues. However, if it is a character forming text, this text is added to the decompression table.

In our example, the file to be processed is written in the place where the decompressed file will be stored. The file to be processed is progressively overwritten by the decompressed text. The files are therefore decompressed from the end, so that the decompressed part does not overwrite a useful part of the compressed file.

Note that the fact that the decompression table is created dynamically is important in terms of memory size since this dynamic table is not added to the compressed file. This table, in fact, contains not only information on the bit pattern redundancy, but also information on the content of the text.

In reference to figure 5, the decompression of the static processing is carried out symmetrically.

According to step E30, the codes of the desired application so converted are received, reassembled, and at least one bit string is replaced by the corresponding byte sequence according to a predetermined conversion function.

A table contains the value of each character converted to enable decompression and regenerate the initial text.

Advantageously, it is planned to combine the static and dynamic processes in the same loop, which considerably reduces the size of the code and the decompression time.

It is possible that the compression may not be efficient with small files (less than 100 bytes) and that the result gives a file of size greater than the initial file. In this case, a tag indicates that decompression is unnecessary and the file is transmitted uncompressed.

Using the invention, it is possible to obtain an average gain of approximately 34% with 0% for a very small file (64 bytes) and 50% for a file of 900 bytes. On average, the dynamic part of the compression results in a gain of approximately 25%, and the remaining 10% result from the static part.

Concerning the decompression, the memory size required is approximately 1100 bytes of which 575 are allocated to storing the character frequency table. The table built in RAM may require 512 bytes.

A detailed description has been given of an example in which the processed data is messages exchanged between a WML/S@Tml page browser installed on a SIM card and a content server located on the Internet. The invention can also be applied to other types of data. For example, the data may also be applets to be downloaded. In this case, the gateway 14 is replaced by a transmitting set which includes an applet server from which the subscriber identification module downloads applets over the air. In this example, the compression according to the invention is implemented directly in the applet server.

A second example of realisation illustrating this solution is as follows. This example illustrates the factorisation mechanism at the very centre of a command. Figure 6 shows a memory block of the CEA memory used to store a service.

This service includes several commands (C1-C11). Each command Cn has its own index and uses a decoding rule specific to the command type. Thus, depending on the user's reaction after executing a command Cx, the execution of this command may lead to various other commands, differentiated by their respective indices. For example, if a command concerns the entry of a PIN code by the user, the next command depends on whether the user enters the code correctly or incorrectly.

Consequently, an index must be stored in each command of the service for each potential future command.

The decoding rules for each command C3, C5, C8 involved in the execution of a particular command C1 must also be included in the byte string of the command C1 to be executed. The number of bytes associated with a command may become very large.

C1 may lead to the execution of command C3, C5, or C8. This command therefore includes the indices of commands C3, C5 and C8 and their respective decoding rules.

We will consider the case, for example, when there are ten command types. Ten different rules are therefore required to decode the command byte string.

In order to illustrate this example, we assume

that the execution of a command may result in the execution of at least 2 different commands from C3, C5 and C8;

and that the total number A of commands does not exceed 25 per service.

According to the principle of the solution, each command is replaced by a reference X (58,35,13) which will both

point directly in a field CH1 to obtain the index (OFFC3,OFFC5,OFFC8) of the next command to be executed, respectively;

and give the type Z of the command to be executed in order to execute the associated decoding rule.

Once the index of the command in question and its type have been found, the next command can be executed.

In our example, this reference is an artificial parameter marked X on Figure 6. Preferably, this reference will have the least possible number of bytes to minimise the space required. It will depend on the maximum number possible for each command type.

The solution consists of using this reference X and of using two separate mathematical functions which can supply two results, one of which gives an index and the other a pointer to a decoding rule.

The range of possible values for this byte is then divided into equal intervals for each command type. A non-limiting way of determining the command types could be as follows:

if the reference X is from 0 to 24 inclusive, the command will be type 1,

if the reference X is from 25 to 49 inclusive, the command will be type 2, and so on.

In our example of realisation, the DIV operator will provide such a result. The command type is obtained by the mathematical operation:

Z = X div A,

i.e., with A=25, Z= X div25

The mathematical operation

Y = X mod A,

i.e., with A= 25, Y = X mod 25

is used to obtain the index of the next command to be executed in the field CH1 defined previously.

As a reminder, the properties of the DIV and MOD operators are as follows:

n DIV p = q: integer division of n by q gives the integer part of the quotient

n MOD p= r: modulo division of n by p gives the remainder r.

CLAIMS

1. A method of loading an application comprising codes onto a subscriber identification module (8), the method comprising:

- a compression step in which each code of a pre-defined set of codes is replaced by a reference to that code so as to obtain a compressed application;
- a loading step in which the compressed application is loaded onto the subscriber identification module;
- a decompression step in which each reference comprised in the compressed application is replaced by the code that is referred to so as to reconstitute the application;
- and a storage step in which the application is stored in a memory of the subscriber identity module.

2. The method according to claim 1, wherein compression includes the followings steps:

- process dynamically at least some byte sequences contained in the said codes so as to recognise in these byte sequences, the byte sequences identically duplicated at least twice,

- for at least one duplicated sequence, define a compression sequence, and store this compression sequence at an address chosen in a compression table, replace at least one duplicated sequence in the original data by the address of the corresponding compression sequence.

3. The method according to claim 1 or 2, characterised in that it also includes, for the subscriber identification module, the following steps:

- receive the codes so compressed from the transmitting set (4, 14), and reassemble them;

- recognise in the message so reassembled the non duplicated byte sequences and the compression sequences;

- store each non duplicated byte sequence in a decompression table at a respective address, and replace at least one compression sequence by the corresponding non duplicated byte sequence located in the decompression table at the address indicated in the compression sequence, and thereby obtain decompressed codes.

4. The method according to claim 1, wherein compression includes the following step:

- process statically at least some byte sequences of the said codes, and convert the said byte sequences into bit strings according to a predetermined conversion rule,
- replace each byte sequence by its corresponding bit string,
- and transmit the codes so converted to the mobile device.

5. The method according to claim 4, characterised in that the bit string length depends on the frequency of the interpreted codes in the application to be processed.

6. The method according to claim 4, characterised in that bit strings are stored in a static table containing the value of each compressed code to enable decompression and regenerating the initial code.

7. A Subscriber identification module being arranged to effect the decompression step and the storage step as claimed in claim 1.

8. A Server being arranged to effect the compression and the loading step as claimed in claim 1.

9- The server according to claim 8, characterised in that it includes processing means (18) able to process dynamically at least some byte sequences contained in the said codes, to recognise in these byte sequences, the byte sequences identically duplicated at least twice, for at least one duplicated sequence, to define a compression sequence, and to store this compression sequence at an address chosen in a table, to replace at least one duplicated sequence in the original message by the address of the corresponding compression sequence.

10. The server according to claim 8, characterised in that the processing means of server (18) are also able to process statically at least some byte sequences of the said interpreted codes, to convert the said byte sequences into bit strings according to a predetermined conversion rule, to replace each byte sequence by its corresponding bit string.

11. A Computer program product for a data processing device, the computer program product including an instruction set which when the instruction set is loaded in the data processing device, makes the data processing device perform the step of decompression and the step of storage as claimed in claim 1.

12. A Computer program product for a data processing device, the computer program product including an instruction set which when the instruction set is loaded in the data processing device, makes the data processing device perform the step compression and the step of loading as claimed in claim 1.

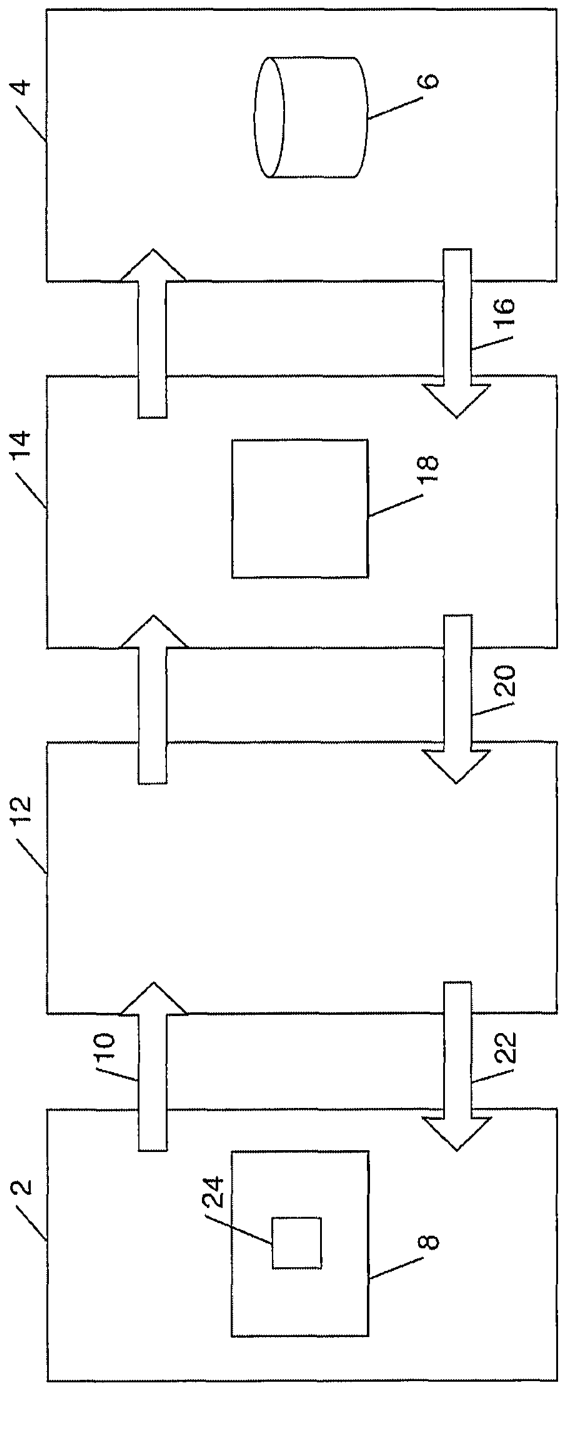


FIG. 1

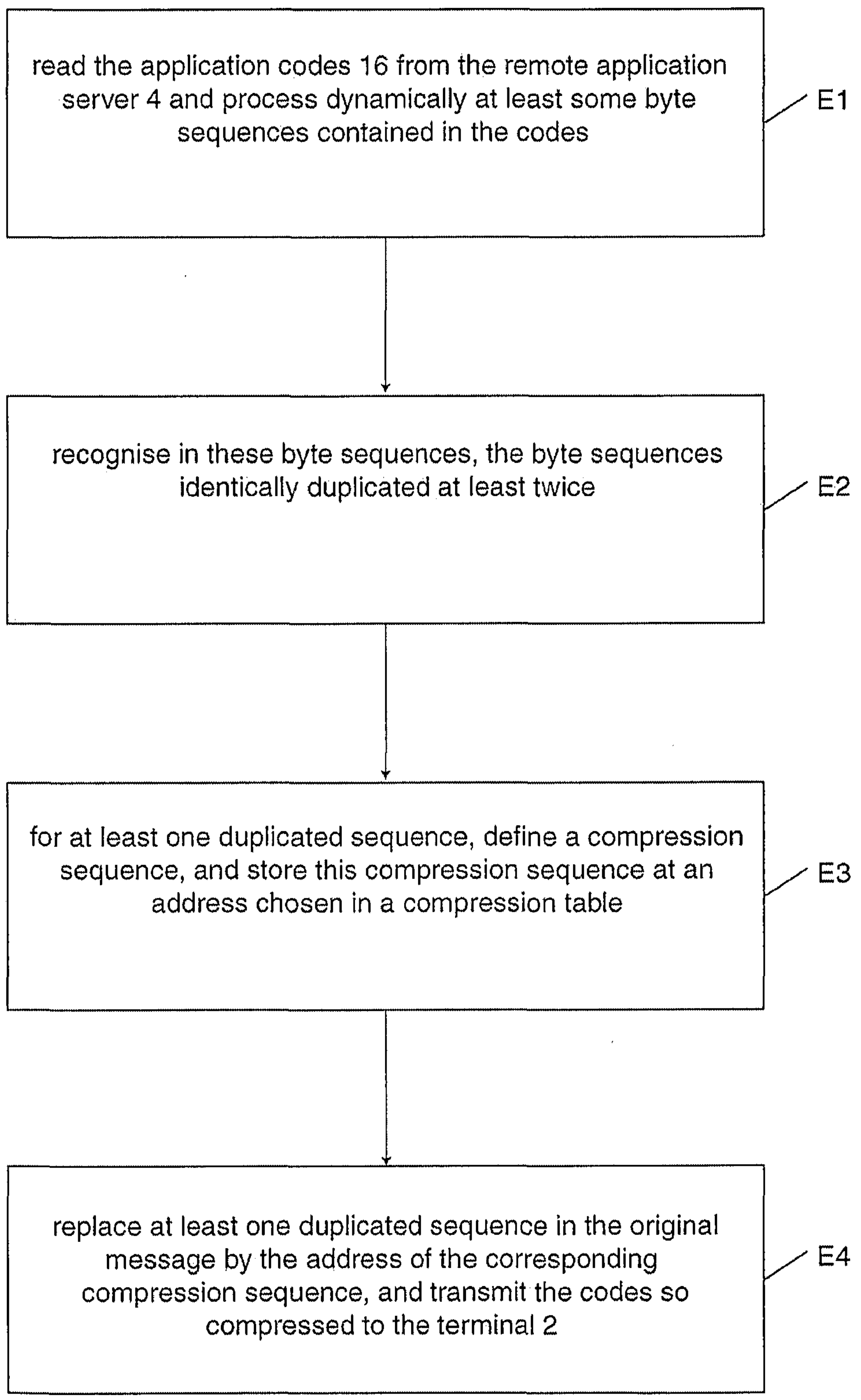

read the application codes 16 from the remote application server 4 and process dynamically at least some byte sequences contained in the codes
E1
recognise in these byte sequences, the byte sequences identically duplicated at least twice
E2
for at least one duplicated sequence, define a compression sequence, and store this compression sequence at an address chosen in a compression table
E3
replace at least one duplicated sequence in the original message by the address of the corresponding compression sequence, and transmit the codes so compressed to the terminal 2
E4

FIG. 2

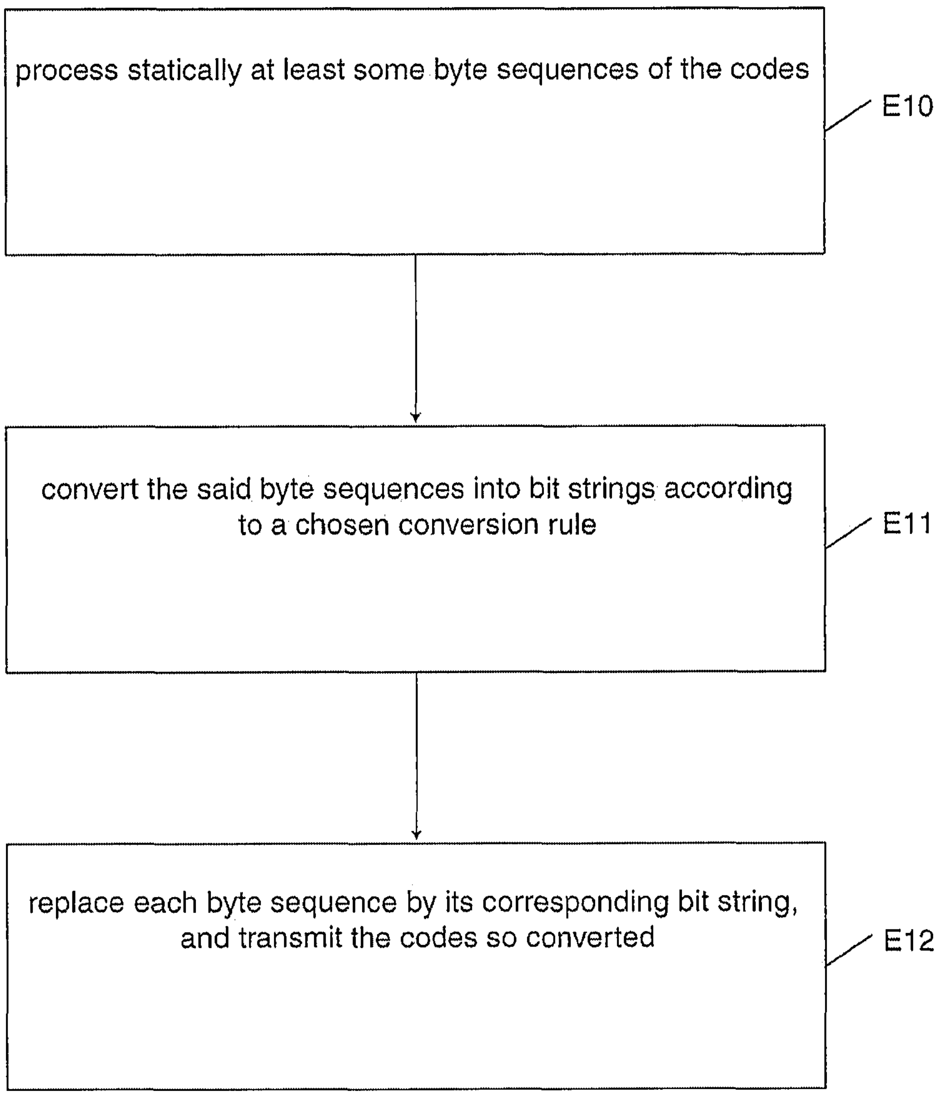

process statically at least some byte sequences of the codes
E10
convert the said byte sequences into bit strings according to a chosen conversion rule
E11
replace each byte sequence by its corresponding bit string, and transmit the codes so converted
E12

FIG. 3

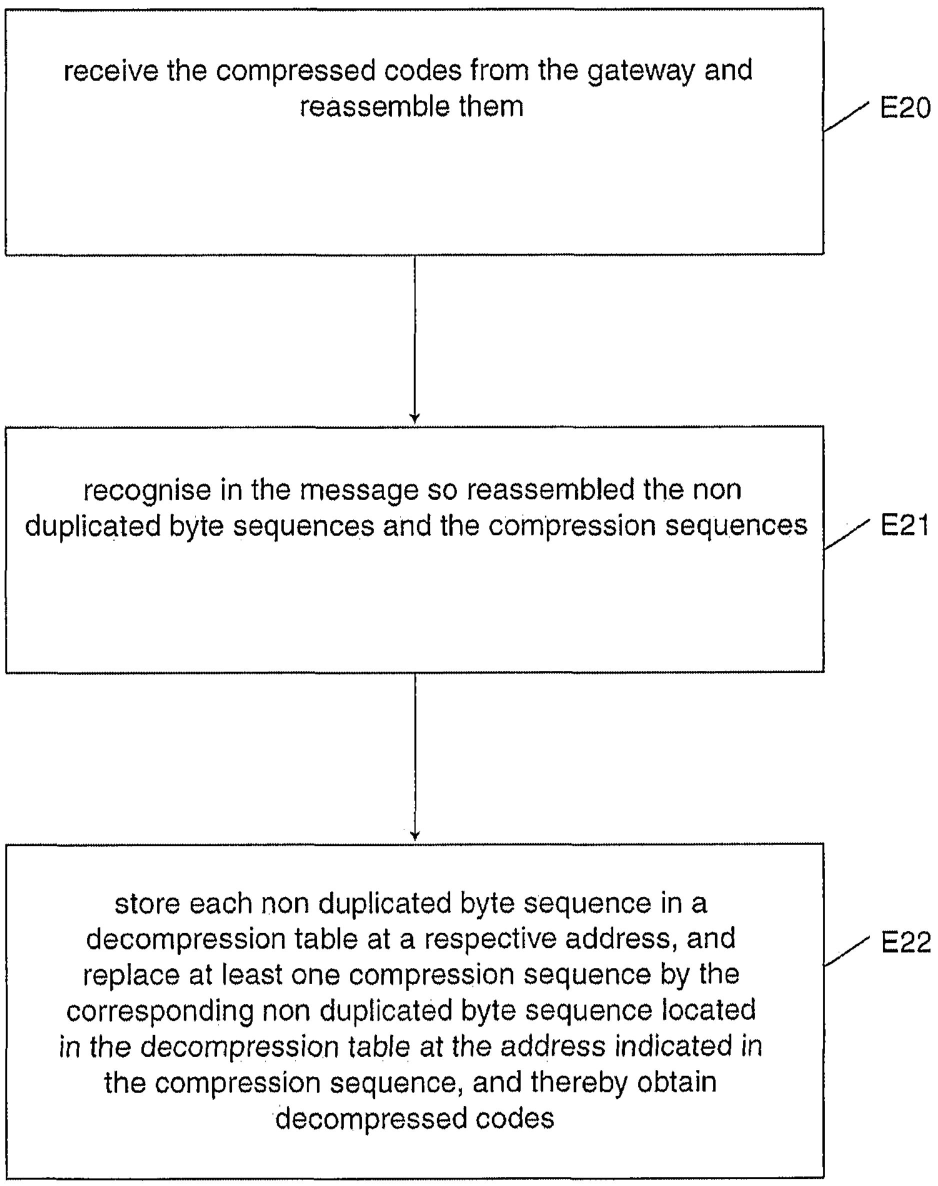
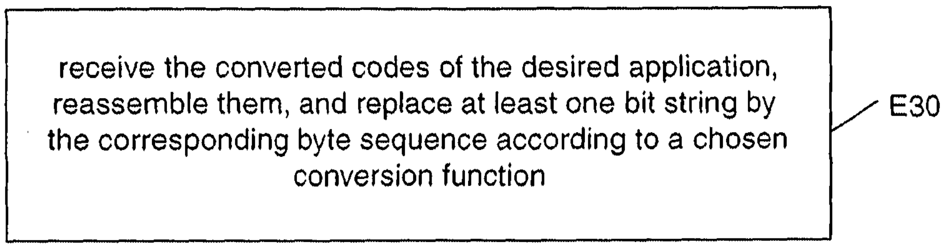

receive the compressed codes from the gateway and reassemble them
E20
recognise in the message so reassembled the non duplicated byte sequences and the compression sequences
E21
store each non duplicated byte sequence in a decompression table at a respective address, and replace at least one compression sequence by the corresponding non duplicated byte sequence located in the decompression table at the address indicated in the compression sequence, and thereby obtain decompressed codes
E22

FIG. 4

FIG. 5


receive the converted codes of the desired application, reassemble them, and replace at least one bit string by the corresponding byte sequence according to a chosen conversion function
E30

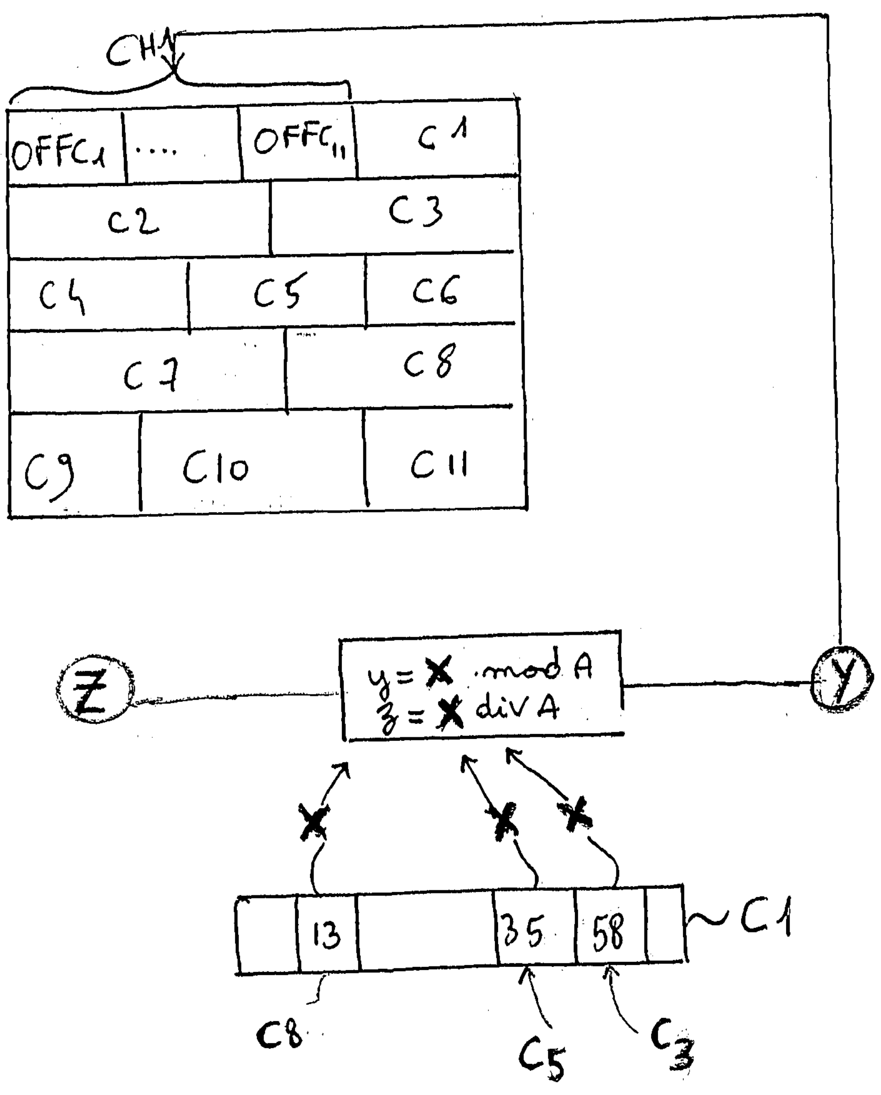

CH1
OFFC1
....
OFFC11
C1
C2
C3
C4
C5
C6
C7
C8
C9
C10
C11
y = X mod A
z = X div A
Z
Y
X
X
X
13
35
58
C1
C8
C5
C3

Fig. 6