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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 August 2006 (31.08.2006)

PCT

(10) International Publication Number
WO 2006/090231 A2

(51) International Patent Classification:
G06F 1/00 (2006.01)

(21) International Application Number:
PCT/IB2006/000344

(22) International Filing Date:
21 February 2006 (21.02.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
05290446.3 25 February 2005 (25.02.2005) EP

(71) Applicant *(for all designated States except US)*: **AXALTO SA** [FR/FR]; 6, rue de la Verrerie, F-92190 Meudon (FR).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **GIRAUD, Nicolas** [FR/FR]; C/O Intellectual Property Dpt, 36-38 Rue De La Princesse, Bp 45, F-78431 Louveciennes (FR). **GOMBOCZ, Pascal** [FR/FR]; C/O Intellectual Property Dpt, 36-38 Rue De La Princesse, Bp 45, F-78431 Louveciennes (FR).

(74) Common Representative: **AXALTO S.A.**; IP & Licensing department, 6, rue de la Verrerie, F-92197 Meudon Cedex (FR).

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

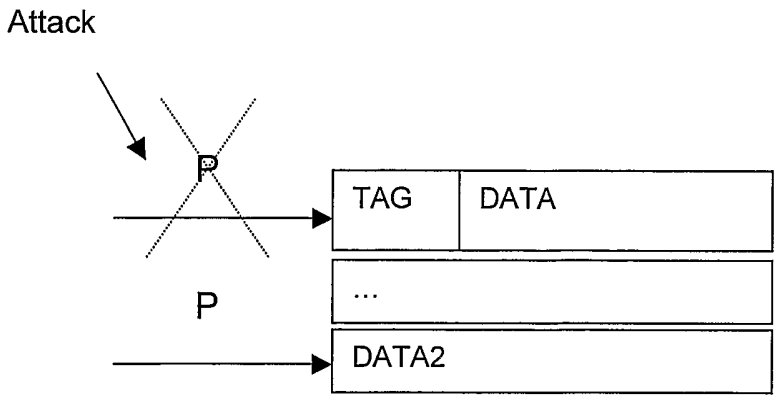
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (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))*

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

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) **Title:** METHOD TO SECURE WRITING IN MEMORY AGAINST ATTACKS BY RADIATION OR OTHER



(57) **Abstract:** The method according to this invention concerns a method to secure an electronic assembly including processing means and storage means (7, 11) containing a program to be executed. The method consists in associating with at least one data item stored in said means (7,11) at least one identification attribute used to classify said data item into a data family. This invention also concerns the electronic module in which said method is implemented and the card comprising said module.

WO 2006/090231 A2

METHOD TO SECURE WRITING IN MEMORY AGAINST ATTACKS BY RADIATION OR OTHER

This invention concerns a method and a device to secure an electronic assembly implementing a program to be protected. More precisely, the purpose of the method is to propose a defence against attacks by radiation, flash, light, laser, glitch or other and more generally against any attack disturbing the execution of the program instructions. These attacks modify the instructions to be executed, the data used and/or the addresses of said data, resulting in non-execution or incorrect execution of certain parts of the program or producing incorrect results.

TECHNICAL FIELD

When a program is executed by a microprocessor, attacks for example by injecting faults via laser, glitch or electromagnetic radiation modify the instruction codes executed by the processor or the addresses of the data to be processed. The program instructions may be replaced by instructions producing a different effect: for example, the attacks may convert any instruction codop into an inoperative instruction code (codop 00h, BRSET0, NOP or AVR depending on the microprocessor). Consequently, certain sections of the code fail to execute or execute irregularly: a security processing sequence in an operating system for smart cards may be made inoperative by an attacker. The attacks may disturb the processor operation and cause untimely jumps in the program memory.

In addition, this type of attack may delete intermediate processing on data used in subsequent processing or modify program pointers to data to be processed. In both cases, the result is that sensitive operations are executed with data other than that planned by the program designers. The fact that a routine is executed with parameters other than those planned may have serious consequences in terms of security. Through this type of

attack, a defrauder could open access to areas of sensitive data, neutralise cryptographic operations and, for example, modify the loading of keys.

One objective of this invention is to propose an efficient defence to avoid executing a program with data other than that planned.

SUMMARY OF THE INVENTION

This invention concerns a method to secure an electronic assembly including processing means and storage means containing a program to be executed, characterised in that it consists in associating with at least one data item stored in said storage means at least one identification attribute used to classify said data item into a data family.

Consequently, when the program is executed, a check is carried out to ensure that the attribute of the data item used by the program corresponds to the planned data item. Otherwise, an attack is detected.

This invention also concerns an electronic module in which said method is implemented, a card comprising said module and a program to implement said method.

BRIEF DESCRIPTION OF THE DRAWINGS

Other purposes, features and advantages of the invention will appear on reading the description which follows of the implementation of the method according to the invention and of a mode of realisation of an electronic assembly designed for this implementation, given as a non-limiting example, and referring to the attached drawings in which:

- figure 1 is a diagrammatic representation of an example of a device in which the method according to this invention is implemented;
- figures 2a and 2b are diagrammatic representations of part of the memory of the device according to this invention in, respectively, the absence and presence of an attack;

- figures 3a, 3b, 3c and 3d represent different steps of a cryptographic process and modifications in memory carried out at each of these steps according to the method subject of this invention.

WAY OF REALISING THE INVENTION

The purpose of the method according to the invention is to secure an electronic assembly and for example a portable object such as a smart card implementing a program. The electronic assembly comprises at least processing means such as a processor and storage means such as a memory. The program to be secured is installed in the memory, for example ROM (Read Only Memory) type, of said assembly.

As a non-limiting example, the electronic assembly described below corresponds to an onboard system comprising an electronic module 1 illustrated on figure 1.This type of module is generally realised as a monolithic integrated electronic microcircuit, or chip, which once physically protected by any known means can be assembled on a portable object such as for example a smart card, microcircuit or integrated circuit card (microprocessor card, etc.) or other card which can be used in various fields.

The electronic module 1 comprises a microprocessor CPU 3 with a two-way connection via an internal bus 5 to a non volatile memory 7 of type ROM, EEPROM (Electrical Erasable Programmable Read Only Memory), Flash, FeRam or other containing the program PRO 9 to be executed, a volatile memory 11 of type RAM, input/output means I/O 13 to communicate with the exterior. In the example illustrated below, the program 9 comprises in particular routines to load data values in buffers in RAM memory 11, one or more computation processes such as a cryptographic computation function (for example a DES function).

The method according to the invention consists in ensuring that the program 9 uses the planned data during execution.

This invention consists in identifying sensitive data of the program by an attribute used to classify the data by a category, type, nature, use phase or other so that the data belongs to a given family and in checking the identification attribute during use. According to the form of realisation described in detail in the following description, the attribute is related to the function which uses the data and/or to the use phase of said data in the program. The attribute identifies the data item with respect to its use. The data items are grouped into categories, classes such as for example input data, output data, etc. (classes corresponding to use phases), and/or the keys, scrambling data, etc. (classes corresponding to a function of the data item). The attribute may also associate a data item with several classes. For example, the attribute associated with a key may correspond to the identifier of this key (MAC or other key): the data item therefore belongs to the class of MAC keys. The key identifier occupies one byte: it is therefore possible to characterise the key on one extra byte. In this case, the attribute belongs to two separate classes. The attribute could be related to any other characteristics of the data, for example the owner.

As shown on figure 2a, the identification attribute (TAG) is associated with the data item in memory (DATA) such that a pointer (P) to this data item points to the set formed by the data item and its attribute. If the pointer is modified (figure 2b), the data item pointed (DATA2) will be identified by another attribute or no attribute at all and incoherence between the expected data item and the data item pointed for the processing will be detected, thereby constituting a security defence.

The cryptographic algorithms may be subject to attacks by injecting faults designed to modify the data processed.

A disturbance on the pointer of the input message of a cryptographic computation function could allow the computation to be carried out with a false message corresponding to data stored in another area of the working memory, possibly an area at 0.The attacker could use this means to obtain a cryptogram with a chosen, or at least known, message.

A disturbance on the key loading operations or on the pointer of the key of a cryptographic computation function could be used to perform the computation with a false key corresponding to a key used in previous processing or to data stored in another area of the working memory. The attacker could use this means to cancel cryptographic operations or obtain a cryptogram with a chosen, or at least known, key.

A disturbance on the pointer of the output message of a cryptographic computation function could be used to avoid storing the result of the cryptographic operation at the place where it is expected by subsequent operations. The attacker could use this means, in some cases, to delete the cryptographic operation from the functional point of view.

In the special form of realisation described below and illustrated in figures 3a to 3d, the invention applies by characterising the memory locations corresponding to the key and to the input and output messages with an attribute corresponding to the cryptographic computation function and taking into account the use of the parameter in the function.

The attribute is for example a "tag" byte prefixing the memory area in which the data item is stored. The pointer on the data points to the memory area containing the "tag" and the data item.

The following values of the attributes associated with the data of the cryptographic computation are defined:

00: invalid data

01: computation input message

03: computation output message

04: cryptographic computation key

Before initialising the parameters of the cryptographic computation function, the memory locations in RAM 11 are characterised by the attribute 00 indicating that no valid data is available. Figure 3a shows the following data items in memory:

DATA 1 is the key K used in the cryptographic algorithm CRYPTO. DATA 2 is the input message INPUT of said algorithm CRYPTO. DATA 3 will contain the result RES obtained by the algorithm.

As shown on figure 3b, when the value DATA1 of the key K has been loaded into the planned location, i.e. in this case loaded into the input buffer memory (buffer in RAM 11) of a cryptography algorithm such as, for example, the DES algorithm, the attribute is initialised with the value 04 indicating that the memory area (buffer in RAM 11) does actually contain a key. The routine to load the key into the input buffer of the cryptography algorithm therefore includes initialisation of the attribute associated with the key to the value planned by said loading routine.

One or more instructions must be added to the software loading routine (loading into RAM or other) of known type in order to assign a value to the attribute of the data item concerned (in this case the key).

Similarly, loading of the value DATA2 of the input message INPUT (figure 3c) is followed by initialisation of the attribute associated with the value 01.The attribute associated with the computation output message is left at value 00 indicating that no valid result is available.

When executing the cryptographic computation function and more especially in the example described when executing the DES software routine, the attributes of the input message and of the key are checked. Said routine includes a check on the attribute values.

One or more instructions must be added to the software routine of the computation process (in this case the DES) or of any other process of known type in order to check the values of the attributes loaded in the buffer by comparison with the values planned in said routine or stored in memory (7).

If the attributes loaded in the buffer do not correspond to the values planned in the routine (04 or 01), a disturbance in program execution is detected and a security defence is triggered. After the computation has been performed (figure 3d), the attributes of the key and of the input message are initialised to 00, then the attribute of the output message is initialised to 02, indicating that the result of the cryptographic computation is available. This attribute will enable the function processing

the result of the computation to check that the memory location does actually store the result of a cryptographic computation.

This mechanism can be used to detect the attacks mentioned above. An attack intended to delete the key loading operation will be detected since the value of the attribute of the memory location which should contain the key will not be 04. An attack intended to modify the key pointer will be detected since the memory area pointed by the modified pointer will have an attribute corresponding to the value present at this memory location instead of 04. As for an attack intended to modify the pointer of the input message, modifying the pointer of the output message will initialise the attribute to a value other than 01 in an area different from that reserved for this data item. The attribute of the memory location containing the real output message will keep the value 00, so the attack will be detected when processing this result.

This mechanism can be extended systematically to the entire software program by associating to each data item an attribute corresponding to the function in which this data item is either an input parameter or an output parameter and by identifying the order number in the list of parameters. The attributes of the input parameters are initialised when initialising the parameters (loading into RAM memory or into a buffer of a computation process) before calling the function (or computation process) and checked at the start of the function (the DES cryptographic function in the example illustrated above). The attributes of the output parameters are initialised before the function return and checked after the function return. This mechanism can therefore be implemented in a compiler.

To implement this mechanism, the code to initialise the attribute of the parameter to be protected and the code to check the attribute must be added. An additional byte is required in RAM memory for each data item to be protected.

According to another form of realisation, the value of the attribute is checked by a cryptographic hardware module, for example the DES

hardware module. In this case, there is no need to add code: the hardware module (e.g. DES) checks the value of the attribute.

According to a development of this invention, the attribute is stored in memory (for example in EEPROM memory 7) permanently. The attribute is defined and associated with a data item permanently, for example during the personalisation. The attribute is predetermined and cannot be changed. As an illustration, a key is stored in EEPROM together with the attribute identifying a key 04. When the key is loaded into RAM, the key is accompanied by its attribute. In RAM, the value of the attribute can be erased; in this case, the data item can no longer be used and its value must be obtained from EEPROM to launch a computation process.

CLAIMS

1- Method to secure an electronic assembly including processing means and storage means (7, 11) containing a program to be executed, characterised in that it consists in associating with at least one data item stored in said storage means (7, 11) at least one identification attribute used to classify said data item into a data family.

2-Method according to claim 1, characterised in that it consists in associating with at least one data item stored in said storage means at least one identification attribute used to classify said data item into a data family so as to detect an attack when the attribute does not correspond to the family expected when using said data item in the program.

3- Method according to claim 1 or 2, characterised in that the attribute identifies the data item with respect to its use in the program.

4- Method according to claim 3, characterised in that the attribute is related to the function of said data item in the program or a part of it and/or to the use phase of this data item in said program or a part of it.

5- Method according to one of claims 1 to 4, characterised in that the attribute is stored in said storage means (7,11) with the corresponding data item such that an access in memory to the data item provides access to the corresponding attribute if any.

6- Method to secure an electronic assembly including processing means and storage means (7, 11) containing a program to be executed, characterised in that it consists, during the execution of said program using at least one data item, in checking that one or more identification attributes associated with said data item stored in said storage means and

used to classify said data item into a data family correspond to the expected family(ies).

7- Method according to claim 6, characterised in that it consists, during the execution of said program using at least one data item, in checking that one or more identification attributes associated with said data item stored in said storage means and used to classify said data item into a data family correspond to the expected family(ies) so as to detect an attack when this is not the case.

8- Electronic module including processing means and storage means (7, 11) containing a program to be executed, characterised in that the storage means (7) comprise at least one identification attribute of a data item stored in said means (7) used to classify said data item into a data family and/or in that it comprises means used to associate with at least one data item stored in said storage means (11) at least one identification attribute used to classify said data item into a data family.

9- Card characterised in that it comprises the electronic module according to claim 8.

10- Computer program comprising program code instructions to execute the steps of the method according to one of claims 1 to 7 when said program is run in an electronic module.

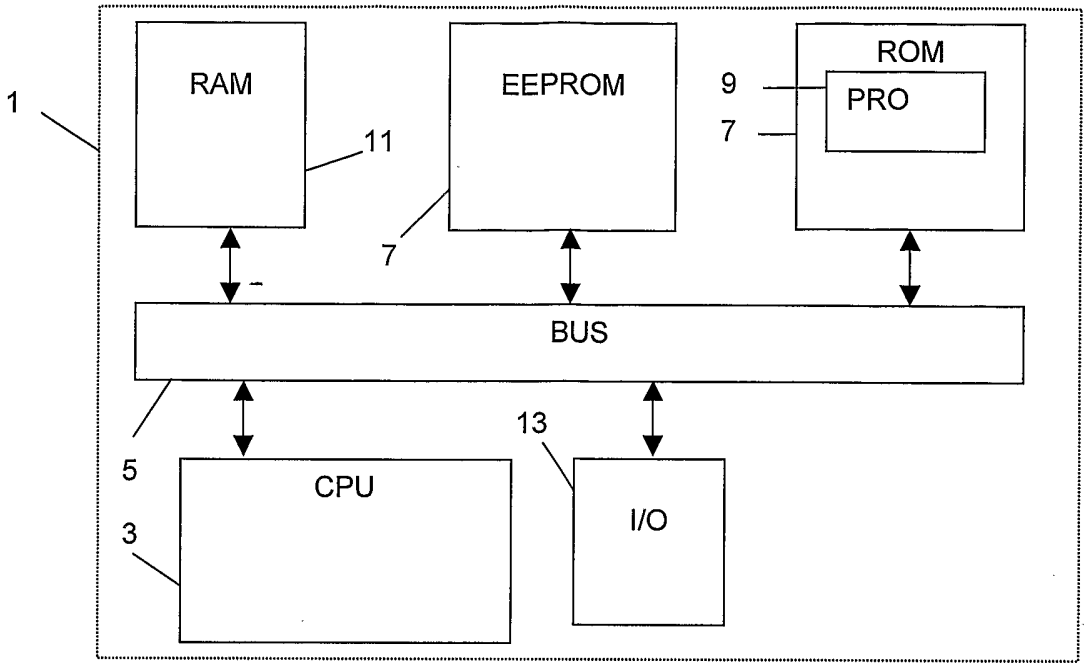


FIG.1

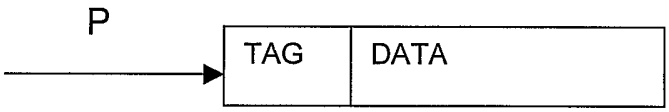


FIG.2a

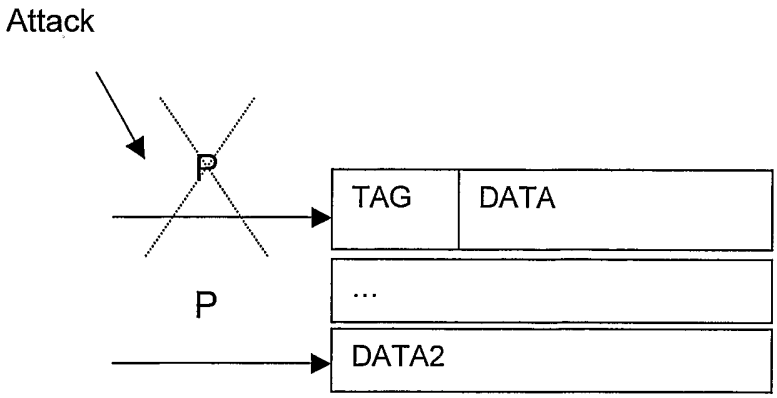
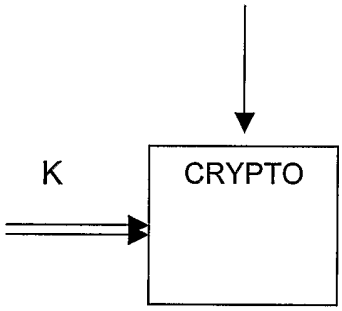


FIG.2b

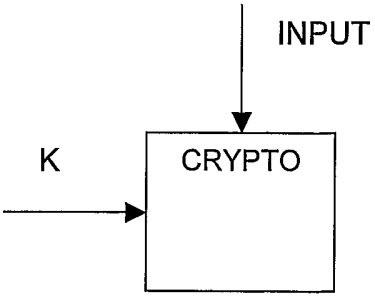
00	DATA1
00	DATA2
00	DATA3

FIG.3a



04	DATA1
00	DATA2
00	DATA3

FIG.3b



04	DATA1
01	DATA2
00	DATA3

FIG.3c

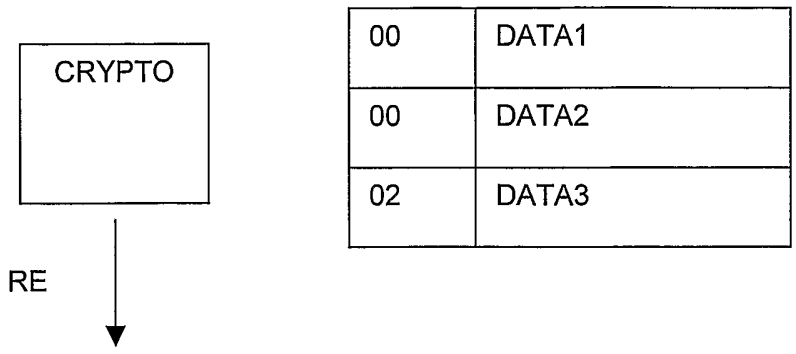


FIG.3d