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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
4 January 2007 (04.01.2007)

PCT

(10) International Publication Number
WO 2007/000207 A1

(51) International Patent Classification:
G06F 21/00 (2006.01)

(21) International Application Number:
PCT/EP2006/004069

(22) International Filing Date: 2 May 2006 (02.05.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
05009410.1 29 April 2005 (29.04.2005) EP

(71) Applicant *(for all designated States except US)*: **ST INCARD S.R.L.** [IT/IT]; Via C. Olivetti, 2, I-20041 Agrate Brianza (IT).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **VARONE, Francesco** [IT/IT]; Via Piero Gobetti, 12, I-81041 Bellona (IT). **VASTANO, Pasquale** [IT/IT]; Traversa Saraceni (p.co. Fellaco), I-81055 S. Maria Capua Vetere (IT). **VENEROSO, Amedeo** [IT/IT]; Via Trentola, 179, I-80056 Ercolano (IT).

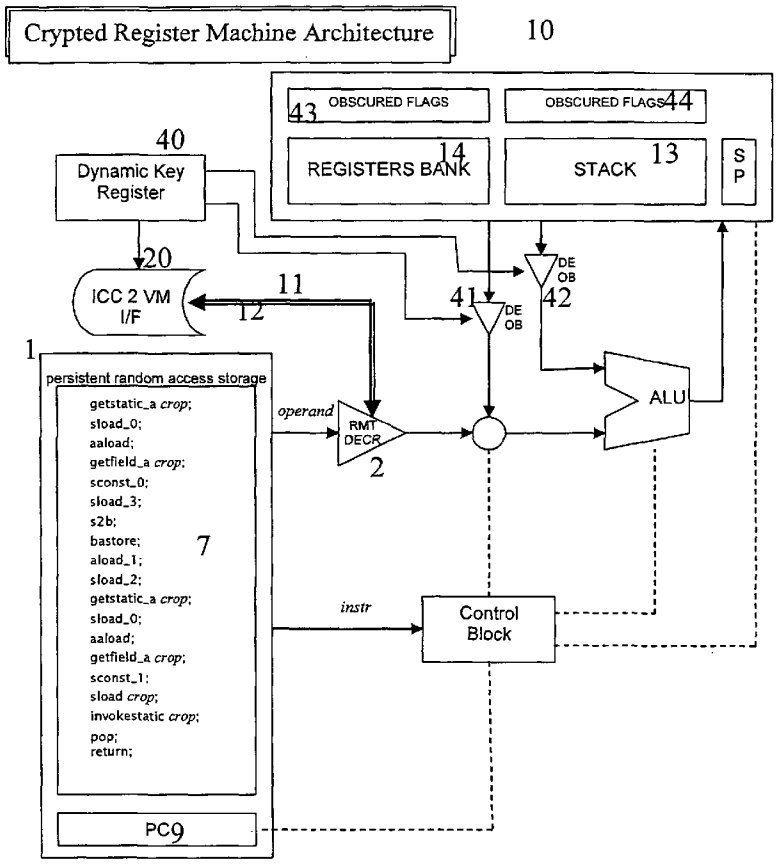
(74) Agents: **BOTTI, Mario** et al.; BOTTI & FERRARI S.r.l., Via Locatelli, 5, I-20124 Milano (IT).

(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).

[Continued on next page]

(54) Title: IMPROVED VIRTUAL MACHINE OR HARDWARE PROCESSOR FOR IC-CARD PORTABLE ELECTRONIC DEVICES



(57) Abstract: A Virtual machine or hardware processor 10 for IC-Card portable electronic devices including a non volatile memory unit 1 a Remote Decryption Unit 2 and associated means for storing executable program in crypted format 7 inside the non volatile memory 1 and for executing them. An IC Card 19 stores a licence Key 8 to crypt and decrypt the executable program through an IC Card Interface 20. The IC Card Interface 20 extracts and crypts only the operands 5 of said plain executable program 4 into crypted operands 6 not to damage the performance. The Remote Decryption Unit 2 detects if an instruction contains crypted operands 6 and queries a decryption 11 to the IC Card Interface 20. The IC Card Interface 20 decrypts the crypted operands 6 and re-crypts the just being decrypted crypted operands (6) into obscured operands 18, through a dynamic obscuration key 17.

WO 2007/000207 A1

Published:
— *with international search 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.

Title: "Improved Virtual machine or hardware processor for IC-Card portable electronic devices"

DESCRIPTION

Field of application

The present invention relates, in its more general aspect, to a virtual machine or hardware processor for IC-Card portable electronic device like digital assistant, smart phones and similar devices, the IC-Card portable electronic device including a non volatile memory unit, storing a plurality of executable programs in crypted format, and a plurality of memory elements, intended to store a plurality of operands derived by the executable programs during execution, the executable programs in crypted format being decrypted during execution by a remote decryption unit in a corresponding decrypted executable programs.

The invention further relates to a method for encrypting and decrypting data in a virtual machine or hardware processor for an IC-Card portable electronic device.

More specifically the present invention relates to a portable electronic system including an IC-Card; an electronic device comprising a virtual machine or a hardware processor and a non volatile memory unit in the electronic device .

Background of the invention

IC-Card portable electronic devices like digital assistant, smart phones and similar devices contain hardware components to store and execute executable program.

In particular a generic hardware architecture is constituted by two main blocks: the first one comprising a non volatile memory unit and a program counter and the second one comprising data memory, registers and stack, as schematically shown in figure 1.

IC-Card portable electronic devices are oftentimes based on a virtual machine architecture. The virtual machine architecture is implemented on the top of a hardware architecture and it is similar to the one described above for a generic hardware architecture. Few difference can be noted: the virtual machine architecture is stack based, so that it has no general purpose registers. Anyway, as stated for a generic hardware architecture, also the virtual machine architecture stores executable program inside the non volatile memory unit.

The virtual machine architecture provides a layer of abstraction between the compiled program and the underlying hardware architecture and operating system, playing a central role in portability.

Non volatile memory unit, not depending on the machine architecture, are readable connecting the IC-Card portable electronic devices to a PC through specific interface or using diagnostic software released by device manufacturer, generally known as software drivers.

Software drivers can be also downloaded from manufacturer support internet site and, in the worst case, they can be copied and simulated by hackers. In this respect, non volatile memory unit cannot be considered a secure support for storing the plain version of an executable program.

As a matter of fact, executable programs are stored inside the non volatile memory unit in non crypted way because they must be executed.

Executable programs so stored in a plain format are in danger because, potentially, they can be copied and reproduced.

Software providers may want to transmit or download executable programs in secure manner so to prevent the executable programs to be stolen in the transmission channel or across device interface.

A prior art document, the European Patent Application N° EP 1 253 503 relates to the encryption of a source code intended to be executed in a electronic device in a high level programming language. More particularly, this document teaches to improve the security in a communica-

tion between the electronic device and an IC Card by introducing a scrambling-descrambling through an encrypted source code and an external unit in which a decryption key for decrypting the encrypted source code is stored.

A protection of mass data is instead described in the Unites States Patent Application N° US 2003/0163718, provided by mapping a plurality of virtual address onto randomly selected actual address.

Even the improved security cited in the above documents protects the communication between an external unit and an electronic device or the mass data storage, the tracking of the source code is not prevented during its execution inside a virtual or hardware processor of the same electronic device, for example when an operand of the source code is temporary stored inside a register or a stack of the virtual or hardware processor of the electronic device.

A United States Patent application N° US 2004/0136530 describes a method to protect source code and/or generic data, intended to be executed by an electronic device, a crypted data being identified through a corresponding "extra" bit present in each memory cell. In this case, a basic architecture of the electronic device, such as volatile and/or non-volatile memory structure, is modified to host the extra bits and the whole memory is encrypted using an internal keys.

Another prior art document, the United States Patent Application N° US2001/0037450 describers a method of developing a protected software application comprising identifying segments according to a protected instruction set. More particularly, a portion of a source code, coded in a first language and intended to be executed by an electronic device, is compiled in a second language and is decoded and executed in such second language by a processing unit external to the electronic device. External processing unit is usually much slower and more limited than the electronic device. Moreover, implementing such solution is usually expensive because it requires an additional external unit provided with higher computational power with respect to the power of the electronic device, in order to improve security. This docu-

ment also specifies that the source code in the first language is completely recompiled and it is protected by an asymmetric cryptography.

In such solution, even if the source code is protected, the execution is delayed because the entire portion of the source code must be recompiled. The asymmetric cryptography is slower than an symmetric cryptography and introduces an additionally delay. Moreover, also this prior art document does not prevent the tracking of a source code when its operands are temporary stored inside a register or a stack.

The technical problem underlying the present invention is that of providing executable programs to be stored in a secure manner inside the memory units of an electronic device, non only encrypting such executable programs inside such memory units but also encrypting operands of such executable program when they are stored inside a stack and/or registers of the electronic device, even if stored temporarily, at the same time without interfering with theeasy execution or the portability of the machine architecture and without limiting the performance with time consuming recompilation or encryption of an entire block of the executable program, substantially without changing the electronic device architecture and overcoming the drawbacks cited with reference to the prior art.

Summary of the invention

The solution idea of the present invention is based on a virtual machine or hardware processor for an IC Card portable electronic device including a non volatile memory unit that stores one or more executable programs in crypted format, the virtual machine or hardware processor decrypting the executable programs in crypted format and re-encrypting them during execution, guaranteeing that also their temporary storing inside the processor registers is crypted, the encryption being executed contemporarily on a set of operands.

According to such a solution idea a first embodiment of the present invention relates to a virtual machine or hardware processor for IC-Card portable electronic devices as defined in claim 1.

Another embodiment of the invention relates to the method defined in claim 20.

The features and the advantages of the machine architecture and of the encryption method according to the present invention will be apparent from the following description of an embodiment thereof, made with reference to the annexed drawings, given for indicative and non-limiting purpose.

Brief description of the drawings

- Figure 1 shows a simplified and schematic representation of a generic machine architecture, according to the prior art.

- Figure 2 shows, in a simplified and schematic representation, a crypted register machine architecture including a Remote Decription Unit, according to the present invention.

- Figure 3 shows a simplified and schematic representation of the executable program encryption.

- Figure 4 represents the IC Card Interface decryption of crypted operands (through licence key) and the subsequent encryption into obscured operands (through a dynamic key).

- Figure 5 shows schematically the obscuration of crypted operands using the Diffie-Hellman key exchange protocol, according to the present invention.

- Figure 6 lists the six step performed by the security system according to the present invention.

- Figure 7 shows an hardware version of the invention.

- Figure 8 shows some alternate configurations of the described architecture. Relevant components with respect to figure 2 are not greyed.

- Figure 9 shows some alternate configurations of the described architecture. Relevant components with respect to figure 2 are not greyed.

- Figure 10 shows some alternate configurations of the described architecture. Relevant components with respect to figure 2 are not greyed.

- Figure 11 schematically represents the collection of many operands to be encrypted in one-shot.

- Figure 12 represents an example of caching operands through an obscuration operands LOOK UP CACHE..

Detailed description

With reference to figure 2, a virtual machine or hardware processor for IC-Card portable electronic devices is represented with numeral reference 10.

In particular the virtual machine or hardware processor includes a non volatile memory unit 1 storing a plurality of executable programs in crypted format 7, a program counter 9, a stack 13 and a register bank 14. In figure 2 is also represented an IC Card Interface 20, between the IC Card 19 and the virtual machine or hardware processor 10.

Advantageously according to the present invention, the virtual machine or hardware processor 10 including the non volatile memory unit 1 is characterized by comprising a Remote Decryption Unit 2 and associated means for storing executable program in crypted format 7 in said non volatile memory unit 1 and for re-encrypting and executing said executable program.

The executable programs in crypted format 7 stored inside the non volatile memory unit 1 are encrypted, for example through an encryption algorithm here not considered because conventional.

Advantageously, the present invention prevents the tracking of the executable program in crypted format also during its execution inside the virtual machine or hardware processor, when it is already decrypted by the Remote Decryption Unit 2.

In fact, after the Remote Decryption Unit 2 decrypts the executable program 7 in crypted format, a plurality of means (20, 40, 17) are provided to perform additional encryptions on the decrypted executable program, such additional encryption being intended to allow, inside the register bank 14 or stack 13 of the hardware or software processor, only a loading of encrypted programs or non plain operands.

More particularly, the virtual machine or hardare processor comprises an IC Card 19 storing a licence Key 8 and an IC Card Interface 20 for decrypting the executable program in crypted format 7 into plain executable program 4 through said licence Key 8.

The IC Card Interface 20 decrypts only the operands 6 of executable program in crypted format 7 into decrypted operands 5.

The Remote Decryption Unit 2 detects if an instruction contains crypted operands 6 and queries a decryption 11 to the IC Card Interface 20.

The IC Card Interface 20 decrypts crypted operands 6 through said licence key 8.

A dynamic obscuration key 17 is generated periodically by the virtual machine or hardware processor 10, for example during the start up of every communication session between the IC-card 19 and the Virtual machine or hardware processor 10.

The obscuration key 17 is stored both in the IC Card 19 and in virtual machine or hardware processor 10 and used to crypt the just being decrypted crypted operands 6 into obscured operands 18. The IC Card Interface 20 returns the obscured operands 18 to the Remote Decryption Unit 2.

In this way, when a crypted operand 6 is decrypted it is immediately reencrypted through the dynamic obscuration key 17, so to protect such operand against a potential attack that occurs when it is temporary stored for execution. In other words, the virtual machine or hardware processor according to the present invention protects operands that are temporary stored inside the register bank 14 and/or stored inside the

stack 13, as well as operands stored in any storing device intended to manage such operands during execution.

With reference to Figure 2, a virtual machine or hardware processor 10 according to the present invention is schematically represented with 10.

In particular the non volatile memory unit is indicated with numeral reference 1, the program counter with 9, the register bank with 14 and the stack with 13.

In the same figure 2 is also represented the IC Card Interface 20, interfacing the IC Card 19 with the virtual machine or hardware processor 10; a licence key 8 is stored inside said IC Card 19, as shown in Figure 7.

More in particular the Remote Decryption Unit indicated with 2 is connected with the non volatile memory unit 1 and the IC Card Interface 20.

A Dynamic Key Register block is represented with 40 and connected with two de-obscurator units 41 and 42 and with IC Card Interface 20.

The register bank 14 and the stack 13 are associated with corresponding obscuration flag 43 and 44.

According to the present invention an executable program in crypted format 7 is stored inside said non volatile memory unit 1; the executable program in crypted format 7 is obtained from a crypting phase of a plain executable program 4, here described in a preferred embodiment.

More in particularly, the whole plain executable program 4 is not totally crypted: the operands 5 are recognised and extracted from the plain executable program 4 as shown in figure 3. The way operands 5 are crypted into crypted operands 6 to produce executable program in crypted format 7, is not relevant for the invention and we can assume that a generic server 90 performs this operation through a licence key 8.

For a better understanding, the encryption process is schematically shown in figure 3: the extracted operands 5 are transformed into crypted operands 6. Said crypted operands 6 are re-aggregated with the remaining code belonging to plain executable program 4 into a partially crypted executable program 7. The partially crypted executable program 7 is finally stored inside said non volatile memory unit 1.

Advantageously, the strategy of crypting operands only, allows to not damage the performance of the machine architecture because operands represent a small although vital part of executable code. An estimate on a JavaCard assembly code (like the one shown in figure 2), where the values 0, 1, 2, 3 are often implicit in the operative code, says that explicit operands represent about 10% of the whole code.

The same licence key 8 used from the generic server 90 is stored on the IC-Card 19; it constitutes a secure support for the point of view of the executable code, since it can be transferred from and to the device with no security problem. More in general an IC-Card could also execute many kinds of application but it is much slower than modern smartphones or other similar devices and for that reason the application must reside in device to be efficiently executed.

The executable program in crypted format 7, so stored in non volatile memory unit 1, must be executed from the virtual machine or hardware processor 1. According to the present invention, when an instruction is fetched from the non volatile memory unit 1, the Remote Decryption Unit 2 is responsible to detect if the instruction contains a crypted operand 6 or not. In the first case, the Remote Decryption Unit 2 queries a decryption to the IC Card Interface 20, as schematically shown with 11 in figure 2.

The IC Card Interface 20 decrypts the crypted operand 6 through the licence key 8 stored inside the IC Card 19. The IC Card Interface 20, before returning the values of the operand, perform an additional encryption based on a dynamic obscuration key 17.

Advantageously, this additional encryption prevents the tracking of the operands also during its fetching for execution, for example inside the register bank 14 or inside the stack 13. Such additional encryption is intended to allow, inside the register bank 14 or stack 13 of the hardware or software processor, only a loading of encrypted operands or non plain operands.

The dynamic obscuration key 17 is generated by the virtual machine or hardware processor 10, usually when execution starts, and sent to the IC-Card 19 when needed, for example using the protocol shown in figure 5 (this protocol follow, for instance, the Diffie-Hellman key exchange protocol). As shown in figure 2, the dynamic obscuration key 17 is stored inside a dynamic key register unit 40.

As an alternative, the complexity of the algorithm for generating the dynamic obscuration key 17 may be choosen according to the need of the subsequent obscuration phases.

An additional obscuration is performed to make safer the executable program also during run time. Anyway the obscuration algorithm is preferred to be much weaker than the one used to generate crypted operands 6, for performance reasons. Crypting algorithms quality shall be balanced according to IC-card 19 and channel speed. Data to be crypted, however, are dynamic and their sensitivity is low so to improve intrinsic quality of algorithms. Besides, potential weakness of obscuration algorithm is balanced by randomness of the dynamic obscuration key.

As explained above, the IC Card Interface 20 crypts the just being decrypted crypted operands 6 into obscured operands 18 through said dynamic obscuration key 17 and then returns the obscured operand 18 to the Remote Decryption Unit 2.

When the obscured operand 18 is returned to the virtual machine or hardware processor 10, it is stored inside stack 13 or in register bank 14, and the obscurator flag 43 or 44, associated respectively to the

stack 13 or register bank 14, is marked to remember that the value of the operand isn't plain.

Since registers are obscured, a potential access to a volatile memory doesn't allow an immediate recognition of the real value, even during run time. Obscuration is not performed on aggregate data but on elementary cell, at register level, so that even if the plain value is grabbed, it's difficult to understand its meaning in relevant context.

When the execution of the instruction needs the plain value of the obscured operand 18, the plain value is obtained through the deobscurator unit 41 or 42 , according respectively to the fact that the obscurator flag 43 or 44 has been marked.

Advantageously, the invention not only protects the communication between an IC card electronic device, for example a computer, and the IC Card itself but it also prevent the tracking of the executable program during its execution inside the virtual or hardware processor of such IC Card electronic device. The virtual or hardware processor architecture is reinforced against possible attack during the execution of the executable program.

Anyway, according to the present invention, not only a remote decryption unit is used to decrypt an already encrypted code but additional means 20, 40 with a dynamic obscuration key 17 are provided to perform an additional encryption on the code decrypted by the remote decryption unit.

Such additional encryption is intended to load inside the registers of the hardware or software processor only obscured operands. Also inside the stack of the hardware or software processor are loaded only obscured operands.

To optimise the performance of the virtual machine or hardware processor many operands 5 can be collected and crypted in a single encryption pass. This should enhance quality of encryption and should reduce

the occurrence of the transfers between virtual machine or hardware processor 10 and IC-card 19.

Advantageously, a group of operands 5 may be encrypted through a single transfer between the virtual machine or hardware processor 10 and the IC-card 19. In this way the performance of the virtual machine or hardware processor in the execution of the executable program is not delayed by an encryption of single operators. Moreover, the virtual machine or hardware processor according to the present invention provides that, in a instruction composed of one or more operators and one or more operands, the encryption may be performed only on operands.

The memory units storing the encrypted operands, for example the stack or the registers, are associated to corresponding flag to specify that their contents are encrypted. Advantageously, according to the present invention, an extra bit associated to an encrypted operand is not required to indicate that such operand is encrypted so that the virtual machine or hardware processor does not require changing to the memory units architecture.

As shown in figure 11, a collection of operands 5 shall be made in a predetermined way, that can be repeated in the same way both by the generic server 90 that encrypts the plain executable program 4 and by the system composed of the virtual machine or hardware processor 10 and IC-card 19, that decrypts the executable program in crypted format 7.

For example, the plain executable program 4 can be divided in some blocks 4a, 4b, 4c, 4d of the same size; all operands 5a, 5b, 5c, 5d are extracted by the corresponding blocks and encrypted with the license key 8 into crypted operands 6a, 6b, 6c, 6d. After that, they can be re-aggregated in corresponding blocks of the executable program in crypted format 7a, 7b, 7c, 7d.

As shown in figure 12, when the virtual machine or hardware processor 10 tries to execute a block 7b, it performs a static look-ahead (i.e. not based on logic flow but on the address) of the code in the block, collects

and aggregate crypted operands and send them to the IC-card 19. Data are then returned by the IC-card 19 in obscured form 181, 182. They are stored by the virtual machine or hardware processor 10 in a look-up table 50, each obscured operand 181, 182 associated to the address of the instruction it belongs to (1^{th}, 4^{th}).

In this manner the virtual machine or hardware processor 10 can then locate the proper obscured operand 18, for each instruction i^{th}, during the actual code execution.

When execution continues in a new block 7c, the look-up table 50 is erased and filled in with obscured operands from the new block 7c. The virtual machine or hardware processor 10 knows if the look-up table 50 contains obscured operands 18 that belongs to the current block 7c because it compares the current block number "n" to the number reference "m" of the look up table 50.

The basic mechanism shown in figure 12, for example the address in the first column of the look-up table 50 can be implicit, multiple look-up tables 50 can be used, to store simultaneously the obscured operands 18 that belong to more blocks and so on.

Said machine architecture is eligible to be further specified: in figure 8, 9 and 10 some alternative configurations are schematically shown.

With respect to figure 2, in figure 8 an obscurer block 60 is inserted on the line that carries results from computational unit (ALU) to registers14 or stack13 . This should be useful if the access to registers 14 and/or stack 13 by unauthorized entities is believed probably. In figure 2 in fact, only crypted operands 6, directly loaded from executable program in crypted format 7, can be stored in obscured manner in the registries 14 or stack 13.

In figure 9 a de-obscurer 61 is needed at output of Remote Decryption Unit 2 if operand of instructions different from "load" (e.g. "add" operands and other arithmetic instructions) are allowed to be encrypted. In

fact, both such operands and data retrieved from registers14 or stack 13 shall be in plain form, before feeding the ALU.

Other versions of possible architectures according to the invention can be obtained by combining the structures shown in figure 3, 8, 9.

It is worth to note that the figure 9 is related to a programming model that provide instruction with none or a single operand. Anyway the invention can be extended to programming models where instructions with more than one operand are admitted. Multiple operands can be fetched and processed in a variety of manners (serially, in parallel, by microprograms), but the case is out of the scope of this document.

If registers bank 14 and stack 13 are believed to have a high degree of protection, thanks to intrinsic architecture properties (in particular in a hardware machine), the overall architecture can be simplified by removing Obscured flags 43, 44 and most deobscurer blocks 41, 42, 60, 61 but leaving a de-obscurer block 62 as output to Remote Decrypter Unit 2, as shown in figure 10. In this case executable program is always stored in plain form in registers bank 14 or stack 13.

The same consideration made for a virtual machine, also applies to a hardware machine. Such a machine will be named herein after enCrypted Registers Machine Unit (CRPU). Some differences are expected in this case.

The architecture presented in figure 10 is particularly indicated for hardware context: in this case stack is usually stored in volatile memory unit outside the machine. On the other hand a registers bank, if present, shall be resident inside the machine. Usually, but not always, machine registers banks are more difficult to be read by an external malicious entity, so in some cases, the hardware manufacturer can decide not to include the logic that obscures their contents, such as registers de-obscurers, registers obscured flags and so on. Anyway, this is not recommended since there are some means to read registers, depending on general machine architecture and programming model (an

interrupt service routine –ISR- called at each main program's instruction execution, for example).

In a hardware implementation stack access instructions ("push"/"pop") should be designed to manage extra data that trace the state of obscuring of stack cells. For example, together with each word stored in stack, an extra bit shall be present that specifies if relevant word is obscured or not.

The present invention provides the virtual or harwdare machine 10 a decryption method to execute an executable program in encrypted format 7, stored in non volatile memory unit 1, the decryption method being applied at run-time. Advantageously, the executable program in encrypted format 7 can be always stored on non volatile memory 1 in secure manner and used in plain format only during the execution.

Advantageously, according to the present invention an encryption may be executed with a ciphering algorithm of arbitrarily complexity, for example encrypting only sensible data with a license key in non-volatile memory and obscuring said data, when stored in internal registers or in stack, with an internal dynamic key.

Run-time execution is made safer by obscuring the decrypted code, and de-obscuring it only when needed. The decryption is lightweight and the performance not to damaged thanks to a partially encryption of the plain executable code based on the encryption of only the operands 5 belonging to the plain executable program 4 in crypted operands 6.

CLAIMS

1. Virtual machine or hardware processor (10) for IC-Card portable electronic device, said IC-Card portable electronic device including a non volatile memory unit (1), storing a plurality of executable programs in crypted format (7) and a plurality of memory elements (13, 14), intended to store a plurality of operands derived by said executable programs in crypted format (7) during execution, said executable programs in crypted format (7) being decrypted during execution by a remote decryption unit (2) in a corresponding decrypted executable programs, characterized by the fact that said crypted format is derived by an encryption of a predefined set of data of an executable program in non crypted format and means (20, 40, 41, 42) are provided for re-encrypting said decrypted executable programs in said plurality of derived operands before their storing inside said plurality of memory device (13, 14).

2. Virtual machine or hardware processor according to claim 1 characterized by the fact that more than one operand of said plurality of derived operands is derived by a same encryption through said means (20, 40, 41, 42).

3. Virtual machine or hardware processor according to claim 1 and further including means (43, 44, 60, 61) to perform said re-encryption through an obscuration of operands and de-obscuration of obscured operands (18) of said plurality of derived operands, said re-encryption being more complex with respect to said encryption of a predefined set of data.

4. Virtual machine or hardware processor according to claim 3 characterized by the fact that said means (43, 44) indicate when said memory elements (13, 14) store said obscured operands.

5. Virtual machine or hardware processor according to claim 1, characterized by the fact that said means (20, 40, 41, 42) for executing ex-

ecutable program in crypted format (7) also comprise an IC Card (19) storing a licence Key (8).

6. Virtual machine or hardware processor according to claim 5, characterized by the fact that said means for executing executable program in crypted format (7) comprise an IC Card Interface (20) for reading the IC Card (19) and for decrypting the executable program in crypted format (7) into plain executable program (4).

7. Virtual machine or hardware processor according to claim 6, characterized by the fact that said IC Card Interface (20) is structured to decrypt only a portion of said executable program in crypted format (7) into a corresponding portion of plain executable program (4).

8. Virtual machine or hardware processor according to claim 7, characterized by the fact that said executable program in crypted format (7) comprises a plurality of crypted operands (6) and a plurality of data in plain format, said IC Card Interface (20) being structured to decrypt only crypted operands (6) of said executable program in crypted format (7) into decrypted operands (5).

9. Virtual machine or hardware processor according to claim 8, characterized by the fact that, said remote decryption unit (2) detects if an instruction contains crypted operands (6) and enable a decryption (11) by the IC Card Interface (20) upon detection of one of said crypted operands(6).

10. Virtual machine or hardware processor according to claim 5, characterized by the fact that, said IC Card Interface (20) decrypts crypted operands (6) through said licence key (8) stored inside IC Card (19).

11. Virtual machine or hardware processor according to claim 5, characterized by the fact that a dynamic obscuration key (17) is periodically generated and stored both in said IC Card (19) and in said virtual machine or hardware processor (10).

12. Virtual machine or hardware processor according to claim 11 characterized by the fact that, said IC Card Interface (20) obscures said

decrypted crypted operands(6) into obscured operands (18) through said dynamic obscuration key (17).

13. Virtual machine or hardware processor according to claim 12 characterized by the fact that said derived operands comprises said obscured operands (18).

14. Virtual machine or hardware processor according to claim 12, characterized by the fact that, said IC Card Interface (20) returns said obscured operands (18) to said remote decryption unit (2).

15. Virtual machine or hardware processor according to claim 12, characterized by the fact that, said obscured operand (18) are loaded inside said memory elements (13, 14) and marked by obscurator flags (43, 44).

16. Virtual machine or hardware processor according to claim 15 characterized by the fact that said means comprises a stack (13) and/or a register bank (14).

17. Virtual machine or hardware processor according to claim 12, characterized by the fact that de-obscurator units (41, 42) perform a de-osbcuration operation on said obscured operands (18), through said dynamic obscuration key (17), before sending them to an arithmetic logic unit (ALU).

18. Virtual machine or hardware processor according to claim 12, characterized by the fact that said de-obscurator units (41, 42) prevent the detection of the plain value of said derived operators when they are temporarily stored in a memory unit.

19. Virtual machine or hardware processor according to claim 1, characterized by the fact that additional obscurator and de-obscurator units can be located in different position of the virtual machine or hardware processor architecture to prevent a detection of operands on a flow that starts from the loading of said crypted operands (7) into said remote decryption unit (2) till the processing of a corresponding plain value through an arithmetic logic unit (ALU).

20. Method for encrypting and decrypting data in a virtual machine or hardware processor (10) for an IC-Card portable electronic device, said IC-Card portable electronic device including a non volatile memory unit (1) storing a plurality of executable programs in crypted format (7) and a plurality of memory elements (13, 14) intended to store a plurality of operands derived by said executable programs in crypted format (7) during execution, said executable programs in crypted format (7) being decrypted during execution by a remote decryption unit (2) in a corresponding decrypted executable programs, characterized by

- deriving said crypted format by an encryption of a predefined set of data of an executable program in non crypted format; and

- providing means (20, 40, 41, 42) for re-encrypting said decrypted executable programs in said plurality of derived operands before their storing inside said plurality of memory device (13, 14).

21. Method according to claim 20 wherein more than one operand of said plurality of derived operands is derived by a same encryption through said means (20, 40, 41, 42).

22. Method according to claim 20 wherein further means (43, 44, 60, 61) are provided to perform said re-encryption through an obscuration of operands and de-obscuration of obscured operands (18) of said plurality of derived operands, said re-encryption being more complex with respect to said encryption of a predefined set of data.

23. Method according to claim 22 wherein said further means (43, 44) indicate when said memory elements (13, 14) store said obscured operands.

24. Method according to claim 20 wherein said means (20, 40, 41, 42) for executing executable program in crypted format (7) also comprise an IC Card (19) storing a licence Key (8).

25. Method according to claim 24 wherein that said means for executing executable program in crypted format (7) comprise an IC Card Inter-

face (20) for reading the IC Card (19) and for decrypting the executable program in crypted format (7) into plain executable program (4).

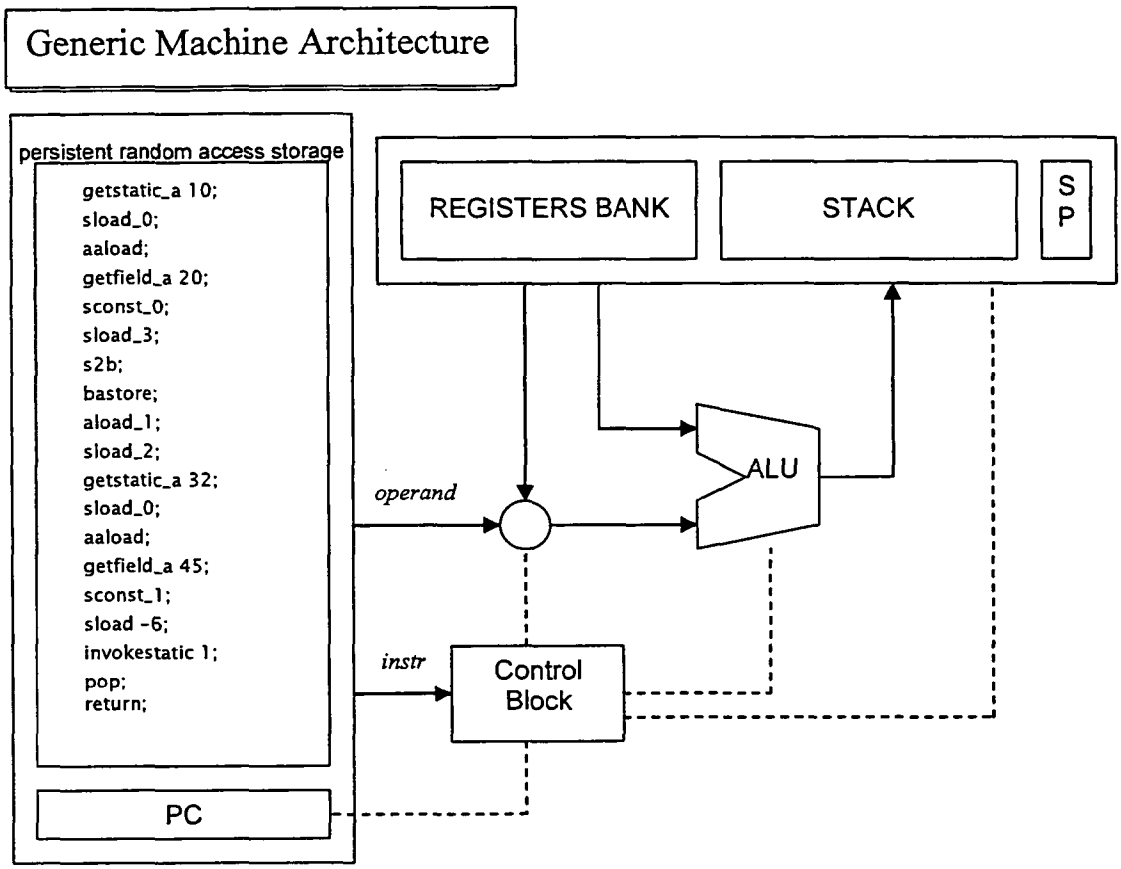

Generic Machine Architecture
persistent random access storage
getstatic_a 10;
sload_0;
aaload;
getfield_a 20;
sconst_0;
sload_3;
s2b;
bastore;
aload_1;
sload_2;
getstatic_a 32;
sload_0;
aaload;
getfield_a 45;
sconst_1;
sload -6;
invokestatic 1;
pop;
return;
PC
REGISTERS BANK
STACK
S
P
ALU
operand
instr
Control
Block

Fig. 1

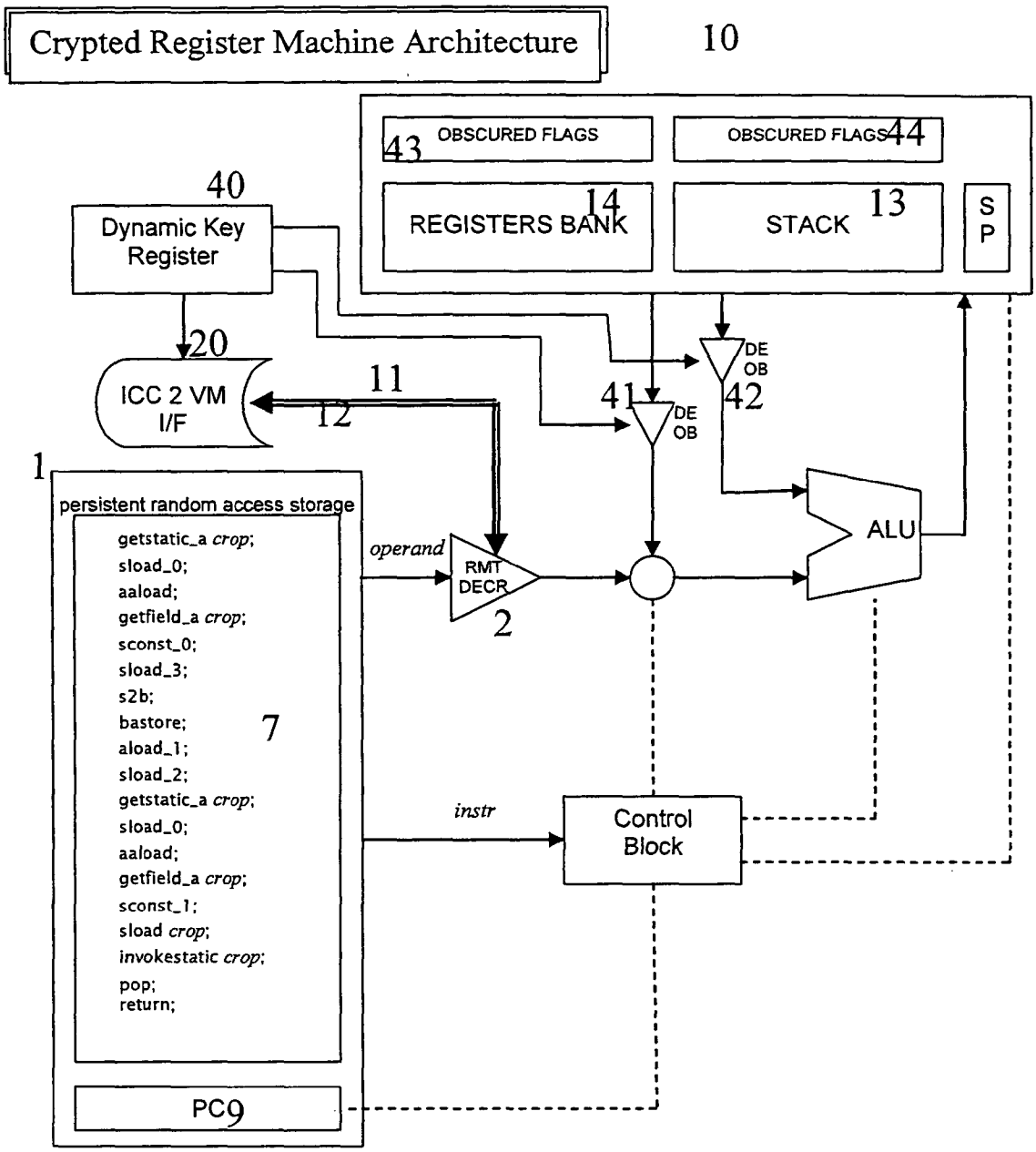

Crypted Register Machine Architecture
10
OBSCURED FLAGS
43
OBSCURED FLAGS
44
40
Dynamic Key Register
REGISTERS BANK
14
STACK
13
S P
20
ICC 2 VM I/F
11
12
41
DE OB
42
DE OB
1
persistent random access storage
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_0;
sload_3;
s2b;
bastore;
aload_1;
sload_2;
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_1;
sload crop;
invokestatic crop;
pop;
return;
7
operand
RMT DECR
2
ALU
instr
Control Block
PC
9

Fig. 2

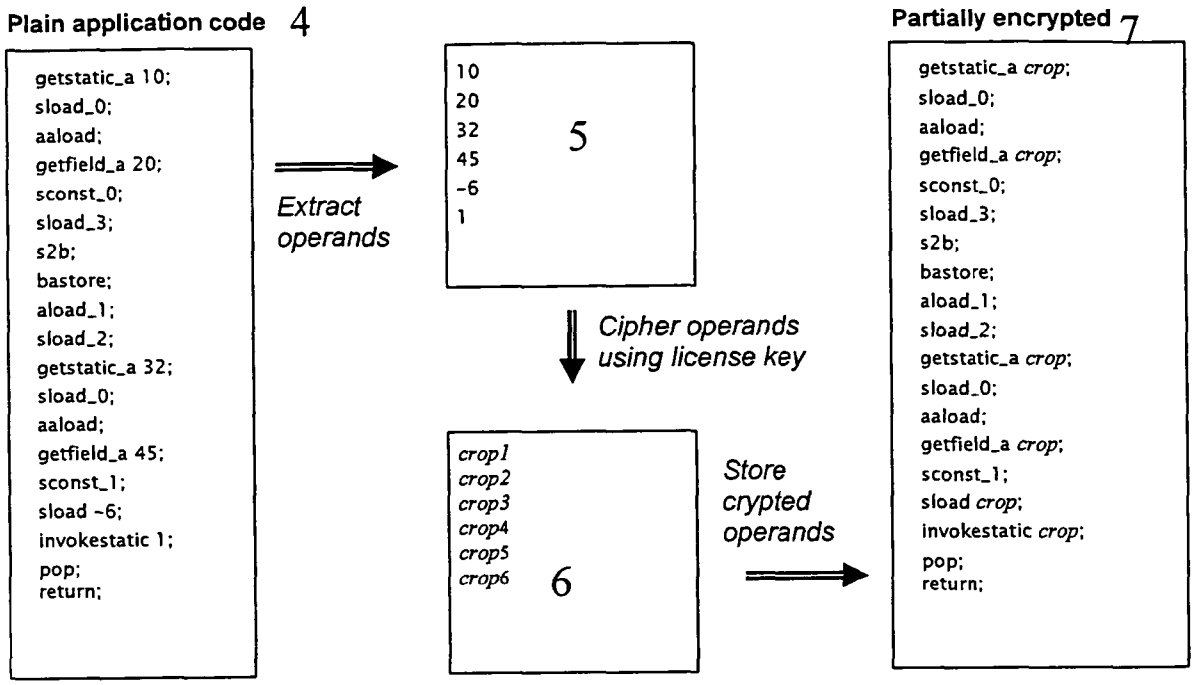

Plain application code 4
getstatic_a 10;
sload_0;
aaload;
getfield_a 20;
sconst_0;
sload_3;
s2b;
bastore;
aload_1;
sload_2;
getstatic_a 32;
sload_0;
aaload;
getfield_a 45;
sconst_1;
sload -6;
invokestatic 1;
pop;
return;
Extract operands
10
20
32
45
-6
1
5
Cipher operands using license key
crop1
crop2
crop3
crop4
crop5
crop6
6
Store crypted operands
Partially encrypted 7
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_0;
sload_3;
s2b;
bastore;
aload_1;
sload_2;
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_1;
sload crop;
invokestatic crop;
pop;
return;

Fig. 3

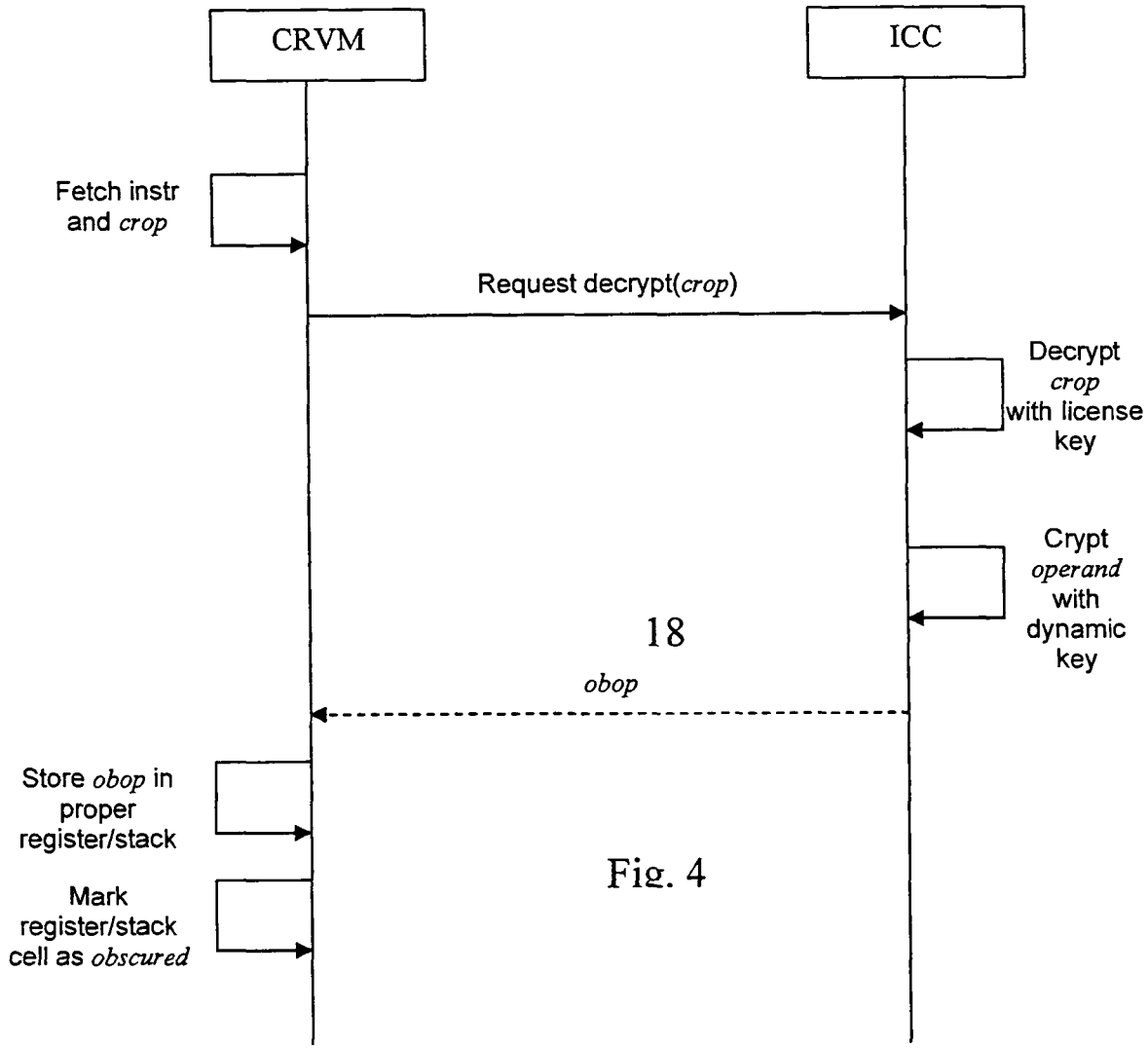

CRVM
ICC
Fetch instr and crop
Request decrypt(crop)
Decrypt crop with license key
Crypt operand with dynamic key
18
obop
Store obop in proper register/stack
Mark register/stack cell as obscured

Fig. 4

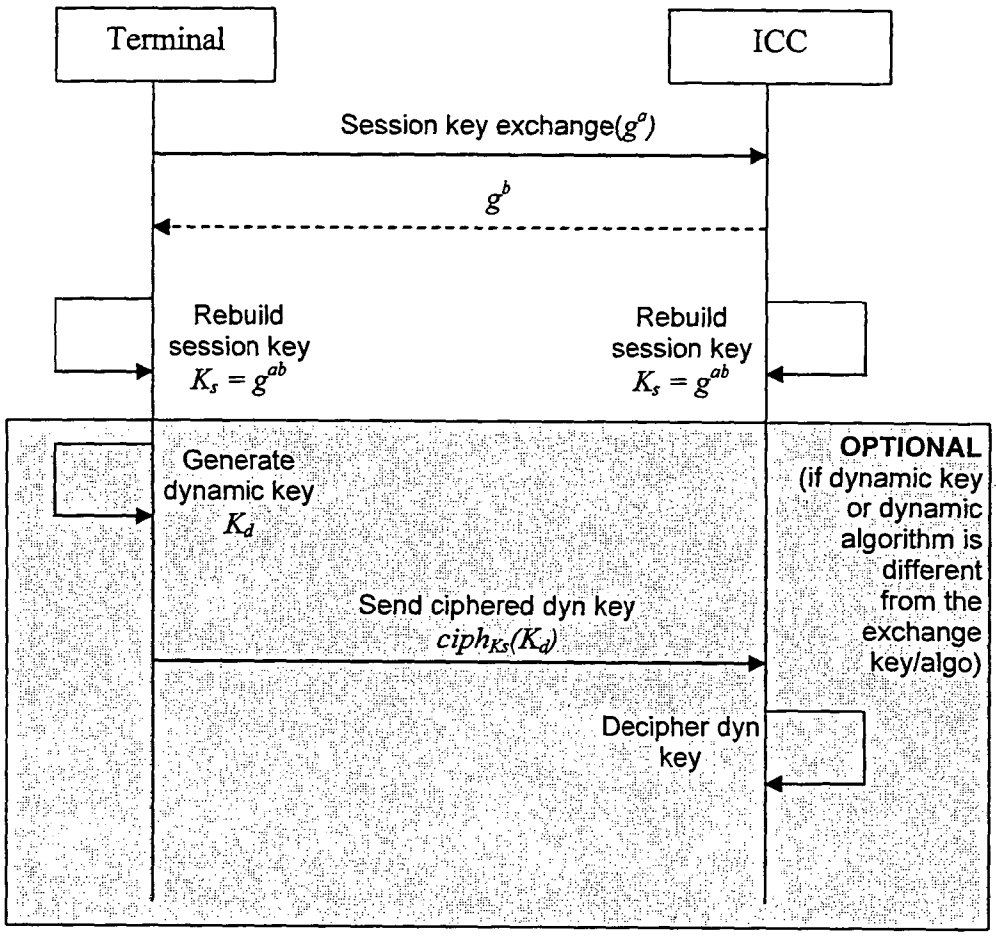


Fig. 5

1. Fetch instruction
2. Decode instruction (Control Block)
3. Fetch operand from registers/stack
4. If cell is marked *obscured*, DEOBs decript values
5. Plain values are processed by ALU
6. Result is stored without *obscuration*.

Fig. 6

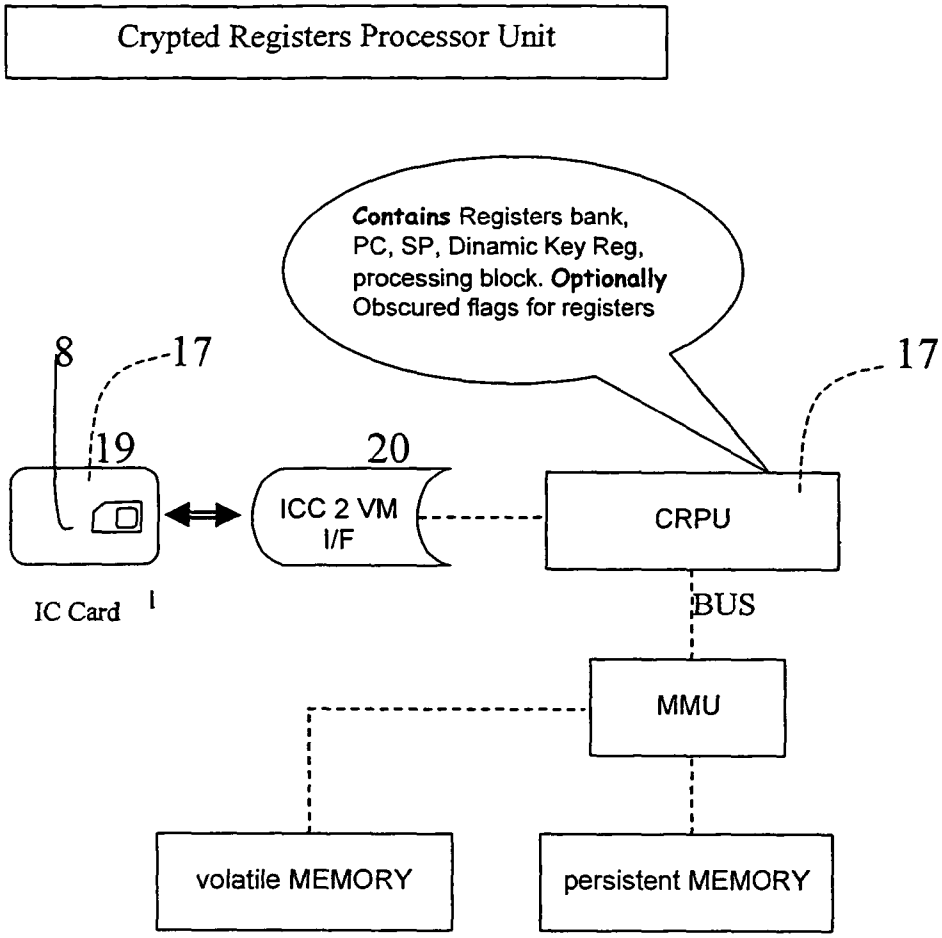

Crypted Registers Processor Unit
Contains Registers bank, PC, SP, Dinamic Key Reg, processing block. Optionally Obscured flags for registers
8
17
19
20
17
ICC 2 VM I/F
CRPU
IC Card
BUS
MMU
volatile MEMORY
persistent MEMORY

Fig. 7

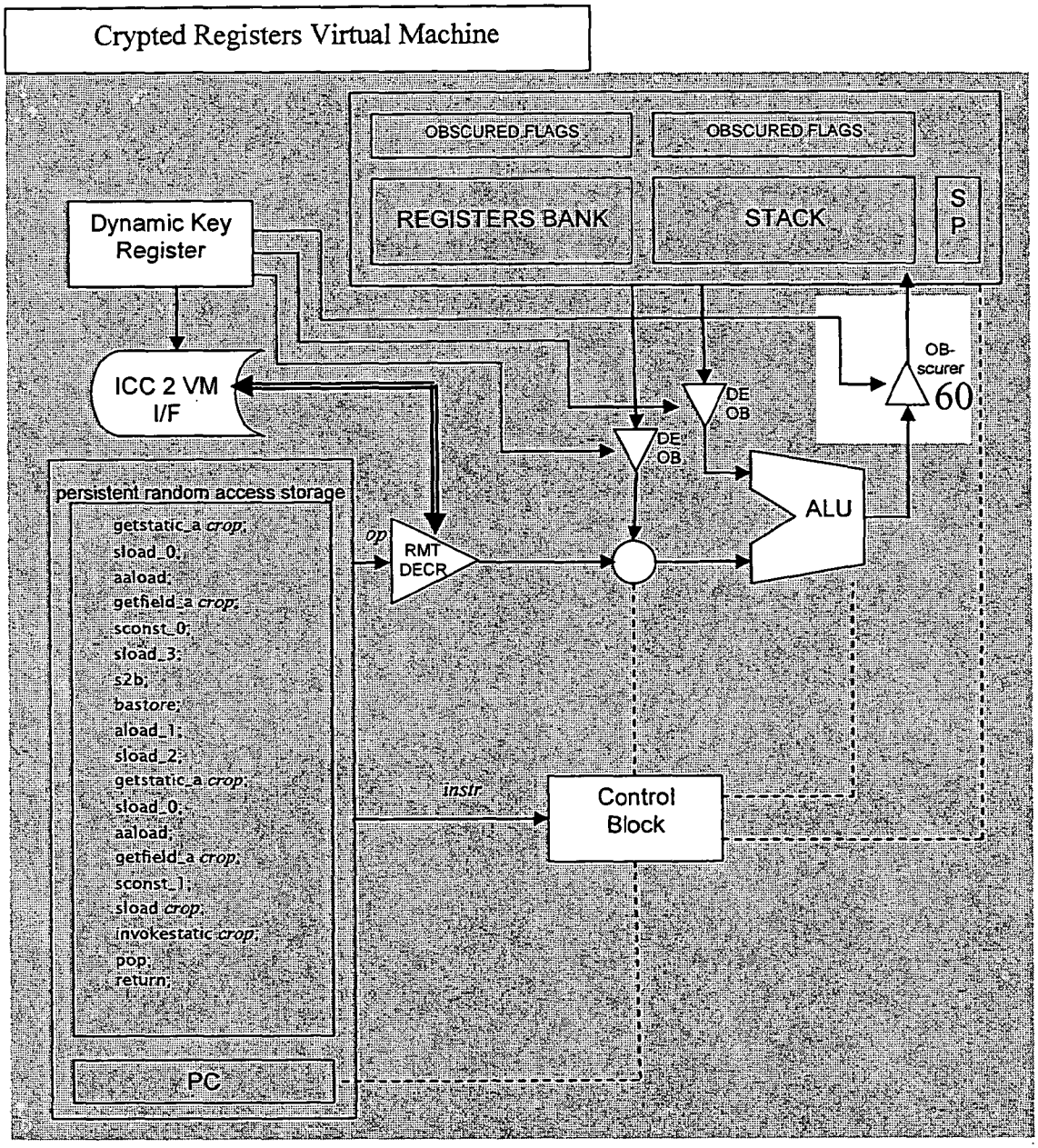

Crypted Registers Virtual Machine
OBSCURED FLAGS
OBSCURED FLAGS
REGISTERS BANK
STACK
S
P
Dynamic Key Register
ICC 2 VM I/F
DE OB
DE OB
OB-scurer
60
ALU
persistent random access storage
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_0;
sload_3;
s2b;
bastore;
aload_1;
sload_2;
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_1;
sload crop;
invokestatic crop;
pop;
return;
op
RMT DECR
instr
Control Block
PC

Fig. 8

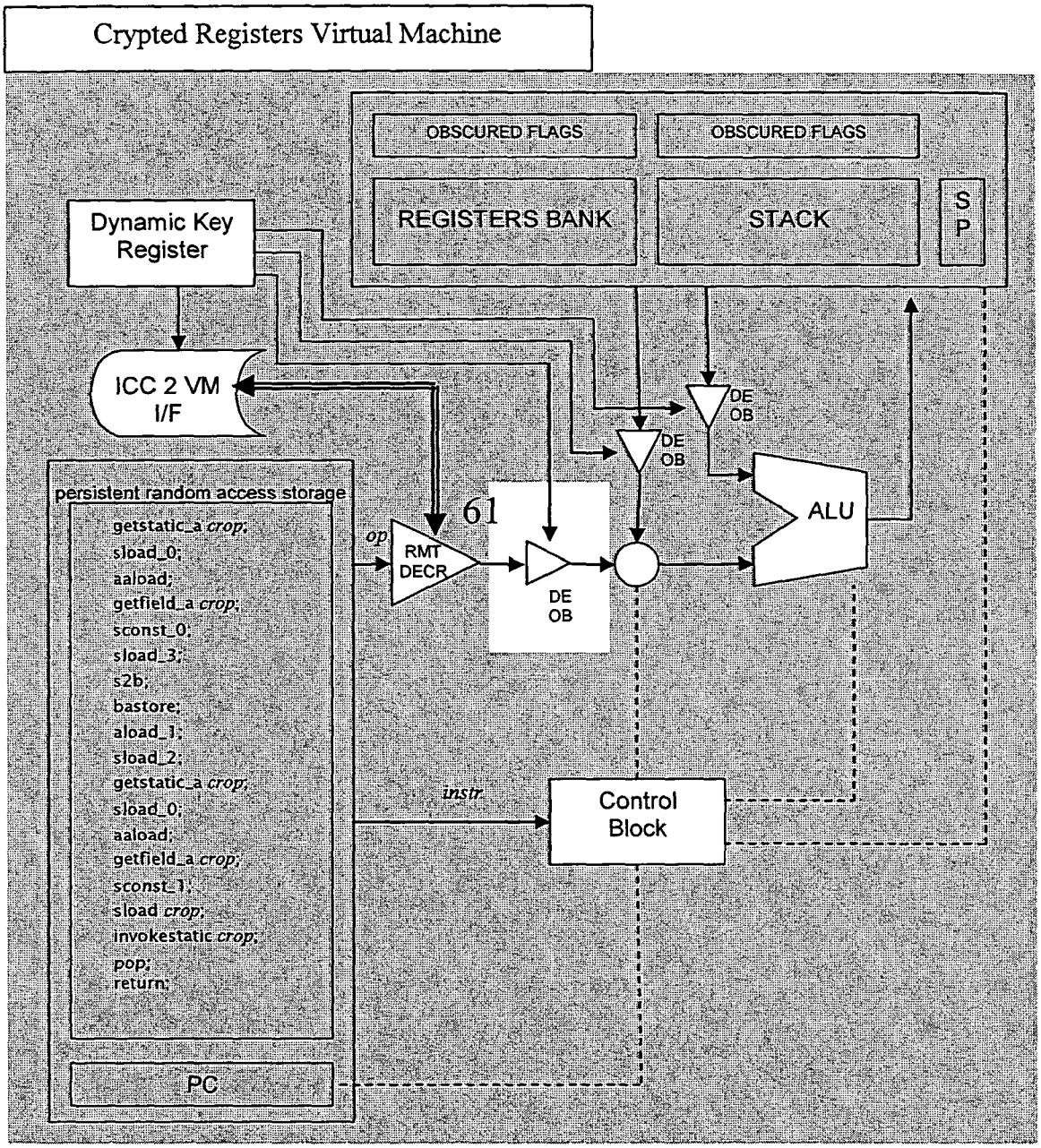

Crypted Registers Virtual Machine
OBSCURED FLAGS
OBSCURED FLAGS
REGISTERS BANK
STACK
S
P
Dynamic Key Register
ICC 2 VM I/F
DE OB
DE OB
ALU
61
op
RMT DECR
DE OB
persistent random access storage
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_0;
sload_3;
s2b;
bastore;
aload_1;
sload_2;
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_1;
sload crop;
invokestatic crop;
pop;
return;
instr
Control Block
PC

Fig 9

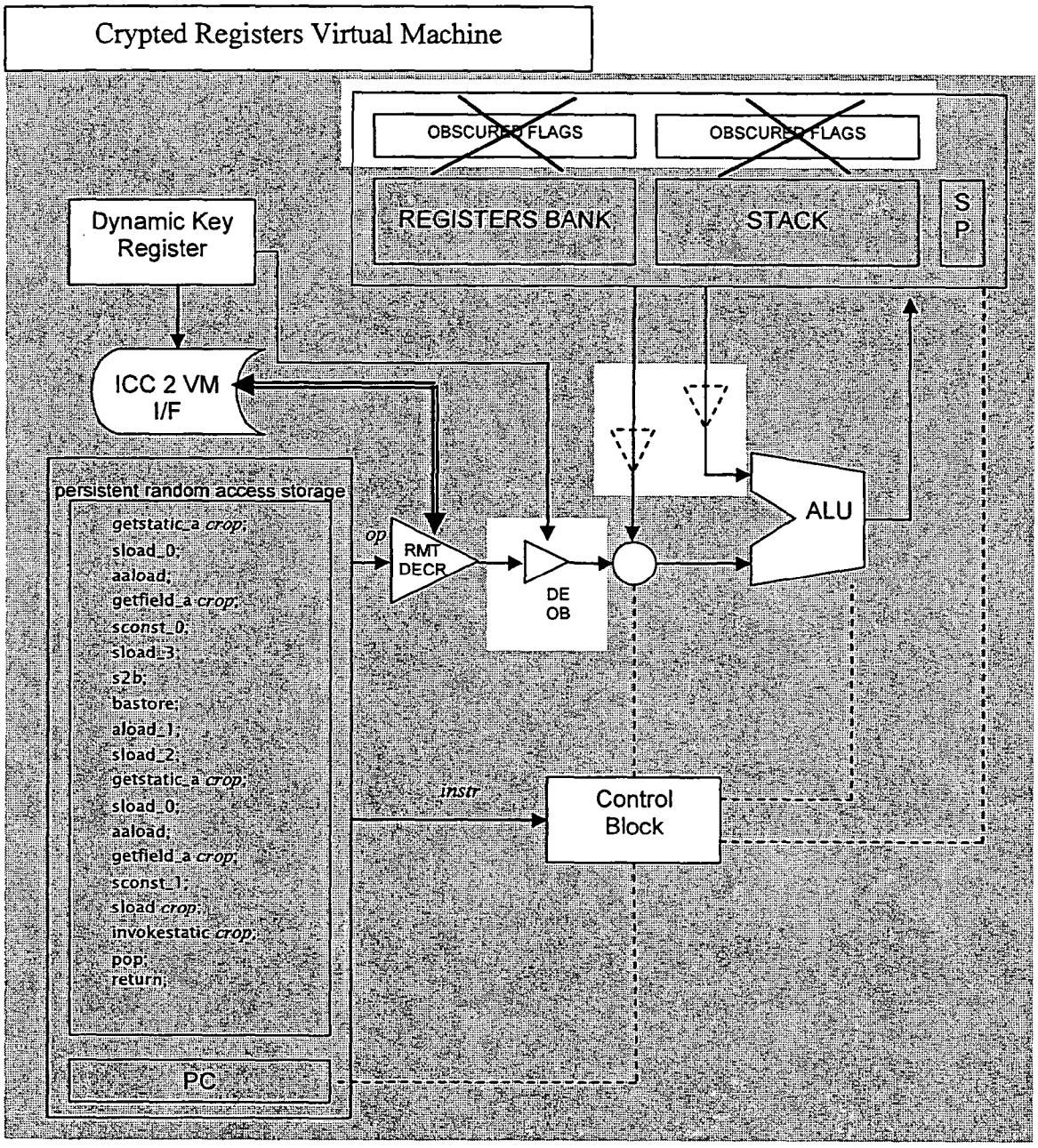

Crypted Registers Virtual Machine
OBSCURED FLAGS
OBSCURED FLAGS
REGISTERS BANK
STACK
S
P
Dynamic Key Register
ICC 2 VM I/F
persistent random access storage
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_0;
sload_3;
s2b;
bastore;
aload_1;
sload_2;
getstatic_a crop;
sload_0;
aaload;
getfield_a crop;
sconst_1;
sload crop;
invokestatic crop;
pop;
return;
PC
op
RMT DECR
DE OB
ALU
instr
Control Block

Fig. 10

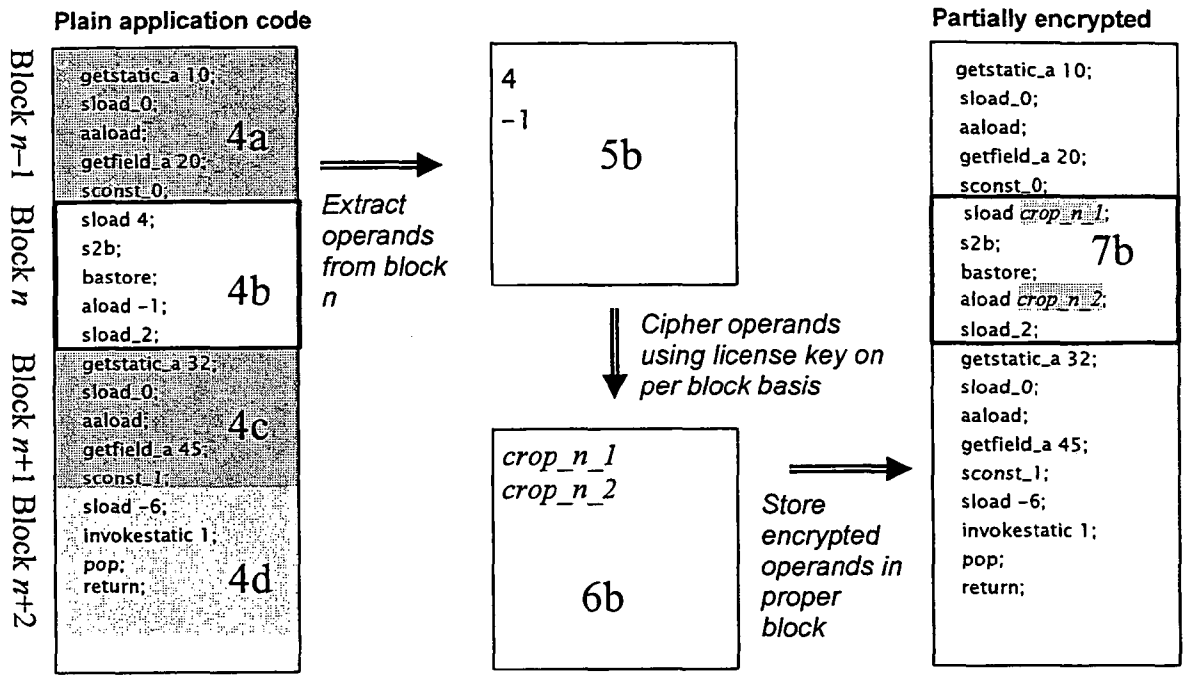

Plain application code
Block n–1
Block n
Block n+1
Block n+2
getstatic_a 10;
sload_0;
aaload;
getfield_a 20;
sconst_0;
4a
sload 4;
s2b;
bastore;
aload -1;
sload_2;
4b
getstatic_a 32;
sload_0;
aaload;
getfield_a 45;
sconst_1;
4c
sload -6;
invokestatic 1;
pop;
return;
4d
Extract operands from block n
4
-1
5b
Cipher operands using license key on per block basis
crop_n_1
crop_n_2
6b
Store encrypted operands in proper block
Partially encrypted
getstatic_a 10;
sload_0;
aaload;
getfield_a 20;
sconst_0;
sload crop_n_1;
s2b;
bastore;
aload crop_n_2;
sload_2;
7b
getstatic_a 32;
sload_0;
aaload;
getfield_a 45;
sconst_1;
sload -6;
invokestatic 1;
pop;
return;

Fig. 11

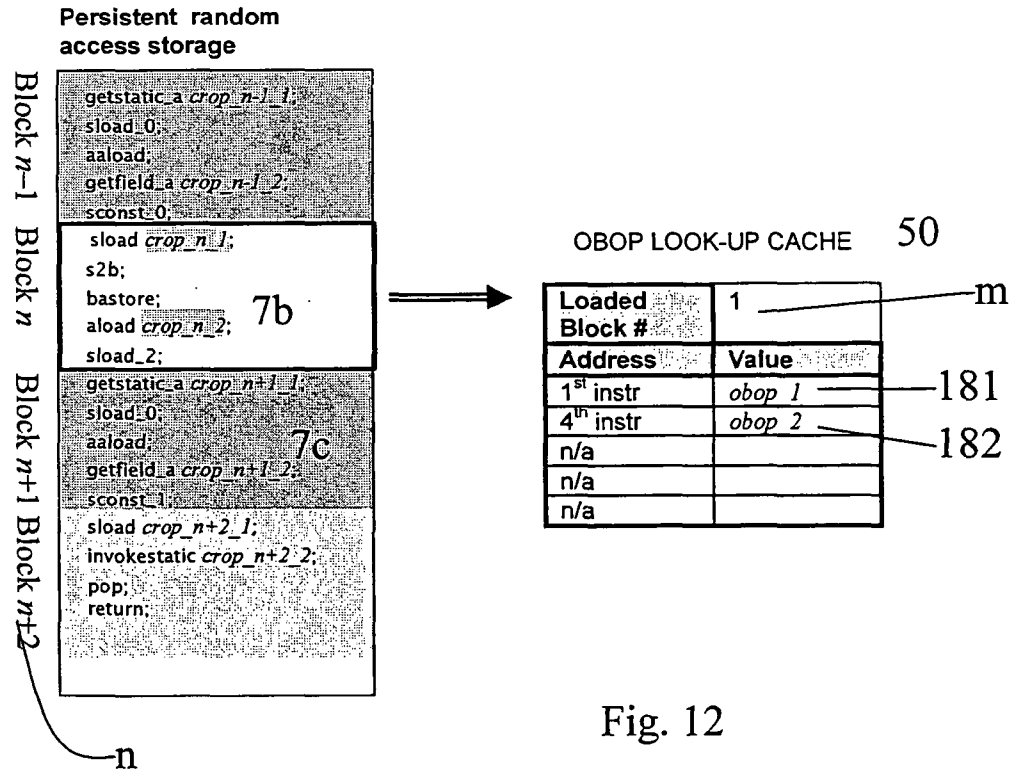

Persistent random access storage
Block n–1
Block n
Block n+1
Block n+2
getstatic_a crop_n-1_1;
sload_0;
aaload;
getfield_a crop_n-1_2;
sconst_0;
sload crop_n_1;
s2b;
bastore;
aload crop_n_2;
sload_2;
7b
getstatic_a crop_n+1_1;
sload_0;
aaload;
getfield_a crop_n+1_2;
sconst_1;
7c
sload crop_n+2_1;
invokestatic crop_n+2_2;
pop;
return;
n
OBOP LOOK-UP CACHE 50
Loaded Block #
1
m
Address
Value
1st instr
obop_1
181
4th instr
obop_2
182
n/a
n/a
n/a

Fig. 12

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/004069

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F21/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 253 503 A (SOSPITA AS) 30 October 2002 (2002-10-30) page 4, line 4 - line 13 page 4, line 50 - page 6, line 21 page 8, line 17 - line 47 page 11, line 49 - line 53 figure 6	1-25
X	EP 1 126 356 A (KABUSHIKI KAISHA TOSHIBA) 22 August 2001 (2001-08-22) column 6, last line - column 7, line 11 page 19, paragraph 220	1,20
X	US 4 558 176 A (ARNOLD ET AL) 10 December 1985 (1985-12-10) column 5, line 26 - line 33 column 6, line 41 - line 60	1,20

-/--

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
31 July 2006	10/08/2006

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Arbutina, L

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2006/004069

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2001/037450 A1 (METLITSKI EVGUENY A ET AL) 1 November 2001 (2001-11-01) page 3, paragraph 30 page 7, paragraph 100 - paragraph 110 page 9, paragraph 130 figures 3,4	1,20
X	US 2004/136530 A1 (ENDO TAKASHI ET AL) 15 July 2004 (2004-07-15) page 1, paragraphs 6,7 page 5, paragraph 73 figures 2,13	1,20
A	US 2003/163718 A1 (JOHNSON HAROLD J ET AL) 28 August 2003 (2003-08-28)	

Form PCT/ISA/210 (continuation of second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2006/004069

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1253503 A	30-10-2002	NONE	
EP 1126356 A	22-08-2001	CN 1309351 A	22-08-2001
		CN 1309355 A	22-08-2001
		EP 1126355 A1	22-08-2001
		US 2001018736 A1	30-08-2001
		US 2001014157 A1	16-08-2001
US 4558176 A	10-12-1985	NONE	
US 2001037450 A1	01-11-2001	NONE	
US 2004136530 A1	15-07-2004	NONE	
US 2003163718 A1	28-08-2003	AU 4818901 A	30-10-2001
		WO 0179969 A2	25-10-2001
		CA 2305078 A1	12-10-2001
		EP 1309905 A2	14-05-2003