

The invention relates to a system for checking the integrity of a piece of software or data, and to a system for implementing this process. It makes it possible to check that information such as data that have to remain constant from one use to another of a data processing system, and that are memorised on a storage medium, in precise locations in that medium, or information constituting the operating instructions for a computer program, also known as software, have not been altered intentionally or accidentally between successive uses. For software to function correctly, its instructions (or any data required between two successive uses) must not have been modified haphazardly.

Indeed, since computers are in almost general use, any user can gain access to application programs, or to the operating systems of the computers.

Because of this possibility of general access, it sometimes happens that users make keying errors that result in an alteration of the program or the data necessary for the operating of the data processing system, or that certain ill-intentioned persons deliberately modify the structure of the programs, or the content of the data in a data processing system, so as to disrupt the operation of the system. In the latter case, the problem is even more crucial than a modification made as a result of an error, since the intentional modification may consist in introducing into the program, or into the data, extraneous elements such as instructions that, when followed by the program, lead to self-modification thereof, which can even cause its complete destruction, by a knock-on effect.

Furthermore, most software is now copy-protected and, for this purpose, contains means that make it possible, when a copy is made, for the copy, or even the original version, to be contaminated and/or to degenerate through successive uses. This can cause a problem when a user has purchased in good faith a copy made fraudulently and quickly sees the software become unusable.

This type of software or data contamination leads to the progressive deterioration of the software and is much more difficult to detect than contamination leading to immediate software malfunctioning. Indeed, this latter type of contamination,

entailing immediate malfunctioning, can be noticed very quickly, since it generally gives aberrant processing results.

In contrast, contamination leading to knock-on deterioration, particularly when done with harmful intent, is not necessarily detectable just by reading the results, since it may be introduced in such a way that the results produced on initial uses are correct or, at least, seem to be, detectable errors becoming apparent after a number of uses.

Finally, after correct software has been loaded with a view to its execution, particularly when the system is operating in a network, a last type of contamination can consist in causing extraneous instructions to be inserted or substituted for some of the normal operating instructions of the software. This can be done remotely, for example via transmission lines.

In this case, even if the user who loaded the software knows that the original software he loaded is correct, he cannot necessarily detect the intrusion and the modification made externally and, as a result, he launches the program which will immediately give aberrant processing results.

Instructions or data implemented in a data processing system are encoded in the form of binary words, each comprising a given number of bits. A usual form is the eight-bit byte, that is, a word contains eight bits, and can have a logic state "0" or "1". Depending on the type of instructions or data, it may be necessary to use several words for any one instruction or any one datum, and a software program is constituted by a succession of these words. The alterations mentioned before can consist either in adding binary words, i.e. extraneous instructions or data, or in modifying the state of one or more bits of certain words in the original program.

A first known solution for detecting alterations, which may prove satisfactory when an instruction or datum has been modified unintentionally, consists in signing the software, that is, one or more binary words following the instructions or data constitute the signature of the software and/or data. To this end, the original

program may be considered to be a message M subject to a transformation $S = f(M)$, the result S of which constitutes a signature which is placed at a given location in the software. For example, the signature can be constituted by the last word of the software.

In order to check the signature when the software is subsequently loaded for use, the signature of the loaded software is recalculated and compared with the signature memorised on the storage medium. When they match, this means that the original has not been altered. Such a process is known, for example, from EP-A-280 035.

However, in the case of intentional fraud, a competent person intending to commit fraud who knows the function or algorithm used to calculate the signature can arrange for the memorised signature to be modified on each illicit intrusion and consequently, at the next use, for the recalculated signature to match the signature most recently stored on the storage medium of the software or data. Thus someone responsible for checking the conformity of a signature will find a match when using the message-conversion algorithm, and will not discover the intrusion.

Furthermore, this method is difficult to apply where the software is on a large scale, since each calculation of a signature for checking purposes requires a certain length of time during which the software is not used for the function for which it is intended.

The aim of the invention is to remedy these disadvantages, by proposing a process and a system making it possible to check quickly and reliably, in all circumstances, that a piece of software or data are in conformity and have not been altered, intentionally or otherwise, between two uses.

The invention is defined by the attached independent claims.

It concerns a process for checking the integrity, against an original message, of a subsequent message containing information, consisting in causing processing means to calculate a signature from the original message to which an algorithm is applied, characterised in that it comprises the steps consisting in:

means to calculate a signature from the original message to which an algorithm is applied, characterised in that it comprises the steps consisting in:

- dividing the original message into a total number of modules;
- using said processing means to apply said algorithm to each module of the original message to calculate a respective signature of each module;
- using an electronic portable object having processing circuits and memorising means including a protected non-volatile memory area accessible only to said processing circuits, and memorising said signatures in the protected memory area and said algorithm in the memorising means;

and checking the subsequent messages in the portable object by the steps consisting in:

- selecting, in the subsequent messages, a partial number of modules less than the total number of modules, said partial number being such that the probability of detecting an alteration in the subsequent message, by considering only said partial number of modules, has a given value;
- using the processing circuits to apply said algorithm to each selected module so as to calculate a respective signature;
- selecting, in the protected memory area, the signatures of the modules of the original message that correspond to the selected modules of the subsequent message and comparing these signatures with those of the subsequent message; and
- causing the portable object to send a response indicating whether the integrity between the original and subsequent messages is confirmed, i.e. whether said signatures of the original message correspond respectively to those of the subsequent message.

The invention is therefore particularly advantageous since it can be implemented at any time during the life of the software or data. Indeed, the memorisation of signatures within a portable object, which can advantageously be a device of the memory-card type with electronic microcircuits, can be performed at any time, by any user who wants to protect himself. This can be done by the designer, in which case the portable object is supplied with the software; it can be performed by a user who wants to check that the software remains uncorrupted during use. In this case

the user obtains a specific blank portable object and runs the program for calculating the signatures he wants, so that they can be memorised, then arbitrarily, using the portable object, he can occasionally check the integrity of the software.

Furthermore, given that the calculation of the signatures with a view to comparing them with the signatures initially stored is performed inside the portable object and since the conformity of the signatures is checked by extracting signatures memorised in an area accessible only to the processing circuits of the portable object, it is impossible for a person intending to commit fraud to deceive the system. It is the circuits of the portable object that recalculate each signature and send them to the comparison circuits, which are accessible only under the control of the processing circuits. Consequently, if the message is corrupted, it is impossible to send a false signature to the address of the comparison circuits, so that any modification or intrusions is detected immediately.

Moreover, the fact that the checking of the message involves not the total number m of modules, but a partial number p less than m , makes it possible to reduce the calculation time required for the checking, with advantageous effect. In so far as the calculation is performed in a portable object, that is, a device having calculation means necessarily limited by the volume available, this reduction in calculation time is valuable.

Other characteristics and advantages of the invention will become apparent from the following description, given with regard to Figures 1 to 6 attached illustrating the principle, and certain embodiments of the present invention.

Figures 1A to 1D illustrate the structure, known in itself, of a series of pieces of information, to which the present invention may apply.

Figure 1A illustrates the typical diagram of a software structure. A software program is composed of a series of instructions, numbered 1 to k in the example, encoded in binary form, using words that can each be located by their addresses. In the example illustrated, the software comprises m binary words, where m can be equal

to or greater than k . Depending on the type of instructions, the encoding of any one instruction can affect several binary words. This has been illustrated in Figure 1A where instruction no. 2 is encoded on two words, words 2 and 3. In a structured data processing system each word generally comprises a given number n of bits, which is generally the size of a byte, that is, eight bits, or a multiple of a byte. Of course, this is not exhaustive and the invention could just as well apply to any other software structure in which the format of the words could vary from one to another.

Software having this structure, with words of the same length, that is, n bits, if taken in a given order, constitutes a message of a total length, expressed in bits, of $L = m \times n$. In general, the total length of the message consists of the sum of all the bits constituting each of the words used for the sequence of instructions.

The invention also applies to checking the integrity of data sequences, as illustrated in Figure 1B. Indeed, it may be necessary to check the integrity of data that may be used repetitively since they are, for example, necessary for running a program.

The data can be independent of the program, as illustrated in Figure 1B showing a set of m words of n bits containing a number j of data.

Like instructions, data can be encoded on several binary words, which explains why the number j of data can be different from the number n of words containing these data, that is, why j can be less than or equal to m .

Figure 1C illustrates a particular example of software in which instructions and data are mixed (x data and y instructions).

It should be noted that, in general, an instruction can be considered to be a particular type of datum.

A piece of software is generally resident in a particular storage medium, for example in the central memory of a computer, or on the hard disk containing the programs of a microcomputer. It can also be used directly from the original medium, such as

intentional or otherwise, results in the modification of one or other of the words mentioned previously.

Figure 1D shows that a word is in fact constituted by a set of bits that can take the value "0" or "1", and the modification of a word results in the change of state of at least one of the bits. Another modification can consist in superimposing extraneous instructions or data on the original instructions or data. This can arise when the software is loaded on a storage medium other than its original medium, for example when the software is installed on a hard disk in order to be used from that hard disk.

When an instruction or a datum, whether extraneous or modified, is read, uncontrollable phenomena can occur.

The present invention consists in considering that the series of instructions or data, which will be denoted in the rest of the description by either of these terms indiscriminately, or by the term information, since the invention applies indiscriminately to checking the integrity of software and/or data, constitutes a message of length L expressed in bits, the value of L being equal to the number of bits constituting the message.

A first solution, applicable directly to short messages, consists in precalculating an electronic signature for the message, using an algorithm A for converting this message, which algorithm can also use at least one secret key, then storing the signature obtained S in the electronic memory of a microcomputer card possessing the algorithm used for precalculating the signature, and the portable object further comprises a memory area accessible only to the processing circuits it incorporates, containing the secret key where this was used when precalculating the signature.

When the integrity of a message is checked the user connects the portable object to the system containing the software, and launches a checking program that causes the processing circuits of the portable object to recalculate another signature involving the series of bits constituting the message to be checked, using the algorithm it contains and, where appropriate, the secret key.

involving the series of bits constituting the message to be checked, using the algorithm it contains and, where appropriate, the secret key.

The recalculated signature is then compared, using the processing circuits of the portable object, with the signature initially calculated and stored in the memory of the portable object, and accessible only to the processing circuits of the latter. If the message used as a basis for calculating the first signature is identical to that used for checking, that is, where the software and/or data have not been altered, then the signature recorded in the memory of the portable object will be identical to the recalculated signature, which can be indicated to the person conducting the check.

Another point of interest in the invention will be immediately apparent: since the signature calculated for checking is calculated by the processing circuits of the portable object and is compared within the portable object by the same processing circuits, it becomes impossible to simulate a false signature, since it would not be taken into account by the processing circuits, in contrast with what happened in the prior art when a person intending to commit fraud could associate with software and/or data a false signature corresponding to the reworked software and/or data.

Figure 2 illustrates the principle of one embodiment of the present invention, with a view to calculating the signature to be stored in the memory of the portable object used subsequently for checking, using a secret key K.

The message M of length L is for example divided into elementary blocks B1, B2, ... Bf, each of which contains a number of bits compatible with the working format of the processing circuits calculating the signature. For example, it could be envisaged that each block contains one bit, but with current processing circuits each block is more likely to be constituted by a number of bits that is a multiple of a byte. The first block B1 can for example be constituted by the first word of the message serving as a basis for the signature, the second block B2 can be constituted by the second word of the message, and so on until the last block Bf which is constituted by the last word of the message. Of course, it is quite possible to envisage complicating the work of a person intending to commit fraud by using a calculation

algorithm that considers the words, or else the bits of each word, in a different order from the one in which they appeared on the storage medium on which they are installed.

The principle implemented consists in calculating a signature, the format of which can be used directly by the processing circuits of the portable object, while the number of bits constituting the message can be significantly greater than the number of bits corresponding to the processing format of the circuits.

The principle implemented, illustrated in Figure 2, consists in performing as many operations as there are blocks, and combining the result of each of these operations, so as to obtain a signature S having a limited number (s) of bits. An initialisation value VI can be applied to the input of the processing circuits involving the algorithm A, and the secret key K is applied to another input of the same circuits, such that the application of the secret key K and the initialisation value VI brings a first intermediate result, which is combined with the content of the first block B1, for example via an EXCLUSIVE OR function. The result of the conversion is produced by means of the EXCLUSIVE OR, which is then applied to a first input of the processing circuits involving the algorithm, the second input taking the secret key K, such that a second intermediate result is obtained at the output of the processing circuits which is combined with the content of the second block B2 through the same conversion function, i.e. the EXCLUSIVE OR.

The same applies to each of the blocks in turn, up to the last one Bf, the content of which is combined through the EXCLUSIVE OR function with the previous result Rf obtained by the processing circuits, by applying the algorithm A to the secret key K and to the result of the preceding EXCLUSIVE OR. The result obtained from the EXCLUSIVE OR applied to the content of the block Bf and to the previous result Rf is combined with the key K through the algorithm A in the signature calculation circuits, and the result of the latter combination constitutes the signature S.

Naturally, the initialisation value VI must be the same as the one that will be used for subsequent calculations, with a view to checking.

When the signature S has been calculated it is then stored in the memory of the portable object, so as to be capable of use as a reference in a subsequent check. When the integrity of the message is to be checked before use, it is then sufficient to get the portable object to check that the signature calculated is indeed equal to the one that has been stored, so that it is inaccessible other than to the processing circuits, in the memory of the portable object.

An initialisation value VI can be constituted by a value contained in the portable object, for example its serial number. It can also be a confidential code, which is entered by the person launching the calculation of the signature that has to be memorised and will subsequently be supplied to other users who may have to check the integrity of the message. Finally, the initialisation value VI can be constituted by the content of a particular memory register of the portable object, the content of which is identical each time the portable object is used. The value VI can also be a random number determined by the processing circuits when the signatures are calculated before being memorised, and which is subsequently memorised at the same time as the signatures.

The embodiment illustrated in Figure 2 is only an example and, naturally, the algorithm can be used in a different way, and both the use of an initialisation value and of the secret key K can be dispensed with, or else functions other than the EXCLUSIVE OR function envisaged can be used.

In the memory of a portable object intended for checking the integrity of a message there is at least one signature S , and the portable object also comprises processing circuits such as a processor and a conversion algorithm A . The portable object is arranged such that the message to be checked can be sent to the processing circuit of the portable object. If necessary, the portable object also contains a secret key K , and in this case a signature is the result of encoding.

Figure 3 illustrates an improved variant of a portable object 1 that contains several signatures $S_1, S_2, \dots, S_m$, and also a unique secret key K in its non-volatile memory area 10, accessible only to the processing circuits 11 of the object. Each signature

is in fact the signature of a different message, and has been obtained from the same secret key K. This configuration can obtain when the same software supplier provides several pieces of software to the same user. In this case the signature of each of the pieces of software can be stored in the same portable object.

Furthermore, the algorithm A is memorised in another area 12 of the object.

This may also be the case when a user who wants protection uses a specific portable object that initially contains a secret key K, and in which he memorises the signatures of each of the pieces of software and/or data storage media available to him.

In such a case, where the same portable object contains the signatures of several messages, each signature has to be associated with means for identifying the original message so that when a signature is checked the processing circuits of the portable object know where in memory to find the original message signature supposed to correspond to the message to be checked. To do this, for example, a serial number or a different identifier is associated with each original message when a signature is precalculated with a view to being memorised. A datum corresponding to this serial number or this identifier is memorised in the portable object at the time that the corresponding signature is memorised, such that the processing circuits of the portable object are in a position to correlate an identification datum with the corresponding signature.

The serial number or identifier can be determined either by the user who wants to precalculate a signature with a view to memorising, or by the processing circuits of the portable object itself.

Naturally, a portable object cannot be used alone and, in this case, as in all variants that will be illustrated subsequently, it must be coupled via coupling and/or interface circuits with a more extensive system 2, especially with the data processing system in which the original message (software or data) is to be exploited. This data processing system is generally part of a computer and comprises at least a

keyboard and printing and/or display means. The coupling and/or interface circuits allow a dialogue to be established between the processing circuits of the portable object and the processing circuits of the more extensive system.

Where it is the user who determines the serial number or identifier of a piece of software, he then enters it in the system, for example via the keyboard of the more extensive processing system. On the other hand, when it is the processing circuits of the portable object that determine the corresponding datum, then the user is informed of the serial number or identifier associated with the original message for which the signature has just been precalculated, following a dialogue that is established between the processing circuits of the portable object and those of the more extensive system, either by the display means of this system or by the printing means.

Whatever the variant used, the user has to keep a record, on a separate storage medium, of the identifier or serial number corresponding to a given original message, and has to indicate it to the system via the keyboard, or other data input means, when a signature is calculated for checking, such that the processing circuits of the portable object will perform the signature comparison solely on the signature supposed to correspond in the memory.

The calculation of a signature, with a view to its memorisation in the memory circuits 10 of a portable object, can be effected directly from the processing circuits 11 of the portable object itself and from the secret key it may contain following manufacture, which is very advantageous, since the signature can be calculated without ever being disclosed externally, for as soon as the processing circuits have finished their calculation they memorise the signature. It can also be envisaged that the first signature calculation, before memorisation, is effected by processing circuits outside the portable object and incorporated in an external system 2 to which the portable object 1 may be connected. The external system 2 is for example a unit for processing the software or data to be checked. These external circuits use the same algorithm as that contained in the portable object. In such a case, if the calculations use a secret key, it may be determined when each signature is

calculated, then memorised at the same time as the latter, or else the secret key can be taken from within the portable object, under the control of the processing circuits it incorporates, then transmitted to the external circuits with a view to calculating the signature before the latter is stored. This solution does however have the disadvantage that the secret key has to be transmitted externally and that it therefore has to be deleted subsequently from the external processing circuits after the signature has been calculated. This solution is however advantageous when voluminous software requires fairly long processing times for calculating signatures. Indeed, the processing time of the central unit of the microprocessor may constitute an obstacle to the use of the process, in so far as the signature is calculated on the basis of the whole message constituting the software and/or data. Indeed, for software of one megabyte it takes more than an hour of calculation to obtain the result of the signature calculation, and hence the result of the check, since the processing circuits incorporated in usual types of portable object, such as microcircuit cards, have processing times distinctly inferior to the processing times of the most powerful computers.

Long processing times during checking may prove absolutely unacceptable for frequent use. For this reason, one variant proposes a checking process that is much faster than the preceding one, and can be used in all cases. For this, the message is first divided into a number of portions or modules $M_1, M_2, \dots, M_m$, and a signature $S_1, S_2, S_3, \dots, S_m$ is associated with each module, and each signature is memorised in a different secret area of the same portable object. Depending on the size of the message for which the signature $S_1, S_2, S_3, \dots, S_m$ is calculated before storage, it is decided whether to have the calculation effected by the processing circuits of the portable object, where the message is not too long, or else, if the size requires too long a processing time, then the calculations are effected by faster processing circuits, for example those of a computer with which the portable object is coupled during the signature calculation phase.

It should however be noted that, in the phase when signatures are calculated with a view to being stored in the memory 10 of the portable object, the calculation time is not critical so that, to avoid disclosing the secret key externally, the solution in

which the calculation is performed by the processing circuits 11 of the portable object itself is preferred.

It will be noted then that the whole message has served as a basis for preparing a number of signatures, such that all the bits constituting the message have been taken into account. In such a case it is possible to use several processes to check the integrity of the message that has served as the basis for preparing several signatures.

A first method consists in causing several p different modules to be chosen at random from among all the m modules that represent the whole of the software to be checked. The number p can be predetermined and constant at each check, and the processing circuits will confine themselves to choosing different modules for a check, being arranged so as to allow for the eventuality of the modules chosen being different from one check to another.

In order to determine the modules, the processing circuits of the portable object use the same process as that used to determine the m modules used for the initial calculation of the m signatures. Thus, if at the time of the initial calculation the message has been divided into modules of k bits, then when the signatures are checked the processing circuits of the portable object again divide the message they receive into modules of k bits, and randomly choose from these modules a number p of modules for which the signature will be checked. The processing circuits of the portable object then calculate the signatures of the p modules chosen and compare them with the signatures that are supposed to correspond in the memory of the card.

The comparison can be made as and when the need arises, that is, when the card has recalculated a signature it checks whether or not this signature matches the signature that is supposed to correspond in the memory, or else all the recalculated signatures can be stored in a buffer memory, then the comparison is made after the p signatures have been recalculated.

As soon as the processing circuits of the portable object detect a difference between a recalculated signature and the signature supposed to correspond in the memory of the portable object, then the message is deemed to be corrupted, and means associated with the processing circuit, such as display means of the system to which the portable object is connected, indicate the positive or negative result of the comparison.

Thus, by way of example, if the original message was divided into eight-bit modules, the first module being constituted by the first eight bits of the message, the second module being constituted by the next eight bits, and so on, the signature recorded for the first module corresponds to the encoding of the first eight bits and the signature for the second module corresponds to the encoding of the ninth to the sixteenth bit of the initial message. If, when the signature is checked, the processing circuits decide to check the signature of the second module, then the processing circuits will take into account the second series of eight bits of the message, the integrity of which has to be checked, recalculate the signature of this second series of eight bits of the message to be checked by applying the algorithm A to this series of eight bits and, if appropriate, by applying a secret key K if it was used at the time of the calculation before memorisation, and the signature stored in the memory of the portable object, corresponding to the second initial module, is compared with the recalculated signature of the module that is supposed to correspond.

Of course, if the p recalculated signatures are correct, although the check is not made on the m signatures initially calculated, then stored in the memory of the portable object, the system nevertheless considers that the software can be deemed not to be corrupted, and the positive result of the comparison is indicated.

The number p of modules for which the portable object has to check the signatures can be predetermined, while the checked modules can differ from one check to another. The number p of modules to be checked is indicated to the processing circuits of the portable object, and it is chosen such that the check is sufficiently thorough to achieve an acceptable level of confidence. Indeed, since p signatures

are checked instead of the m original signatures, the calculation time needed for checking is reduced consistently, in comparison with the calculation time that had been necessary for memorising all the signatures.

Moreover, the number p of modules may not be predetermined in advance, but can be chosen at random by the processing circuits of the portable object. In this case the value of the number p calculated needs to be checked, to avoid the check involving too small a number of modules, such that the validity of the check would not be sufficient. Furthermore, the number p must not be too high, so that processing times nevertheless remain acceptable.

Figure 4 is a curve showing the number p of modules that have to be checked, depending on the total number m of modules contained in the message, to obtain a probability Pr equal to 0.9, i.e. it shows the number of modules that have to be checked in order for there to be nine chances in ten of discovering that the message has been altered in the case where, furthermore, the message contains a module q of altered modules equivalent to the number p of checked modules. It will be observed, then, that for a piece of software containing a thousand modules, around sixty of which were altered, sixty modules will have to be checked in order to have nine out of ten chances of discovering an alteration to a message.

In other words, this means that if around sixty module signatures are checked for a message containing a thousand modules, and if less than sixty modules have been altered, then the probability of discovering an alteration at the end of this sixty checks is greater than 9/10 and, vice versa, it diminishes if the number of altered modules is greater.

In order to determine the number p of modules that have to be checked, a compromise must therefore be made between the number q of modules likely to be altered and the total number n of modules contained in the message.

Assuming that each module can be located by a serial number, as has been stated previously, the random choice of the p modules on which the signature check is to

be made can consist in having the processing circuits of the portable object prepare p different random numbers each determining the serial number of the selected module.

Thus, assuming that four checks are to be made on a set of m modules, the processing circuits of the portable object will four times randomly draw numbers less than or equal to m , and different from each other. If, for example, the calculation gives the numbers 2, 4, j , $m-1$, then the processing circuits of the portable object will calculate the signatures of the second, fourth, j th, $m-1$ st modules of the message, the integrity of which needs to be checked, and will compare them with the second, fourth, j th, $m-1$ st signatures memorised in the memory of the portable object.

Following the comparison, the processing circuits of the portable object will act on the means for indicating whether a difference or a similarity has been detected in the various comparisons.

In a variant, to determine which modules will have to be signature-checked, the processing circuits of the portable object are caused to calculate a binary number (a), of a length m , that is, one comprising a number of bits equal to the number of modules constituting the message. Furthermore, the process ensures that the binary number comes from a code p among m , that is, it comprises a certain number of bits p in a given logic state, while the $m-p$ bits remaining are in a complementary logic state. For example, if the initial state of the bits is the logic state "0", the generation of the number (a) will cause p bits to be in a state different from the rest state. Since each bit can be located by its serial number in the set of m bits, a choice will be made to signature-check the modules, the serial numbers of which correspond to the numbers to the serial numbers of the bits that have changed to "1" in the random binary number (a).

Yet other possibilities can be envisaged by the person skilled in the art without thereby leaving the scope of the present invention, as defined by the claims.

The different variants that have just been presented may prove sufficient and offer a satisfactory degree of security for determining with a high probability whether or not a message has been altered. However, other variants can be envisaged that make it possible to improve the security of the method.

These improved variants are made by taking account of the fact that a fraudulent intrusion generally consists in modifying a series of sequential instructions in the program, or of introducing fairly localised and infrequently-used extraneous sequences. It is only when the program uses these extraneous instructions as it is running that the effects of their presence can prove disastrous. These extraneous instructions may indeed not be called up by the program as it is running under certain conditions of use, while they are under other conditions.

For this reason, in order to increase the probability of detecting an extremely infrequently used local modification, in a variant embodiment, before the signatures are calculated, the algorithm used in the processing circuits of the portable object is such that it organises a division of the message into blocks, irrespective of their content, and organises an interlacing of the blocks in the signature calculations, so as to make any consistent modification of the message impossible. A set of interlaced blocks forms a module.

Thus it may be considered that each elementary binary word constituting the message constitutes a block. In such a case, if the working format of the processing circuits is the byte, each block will be constituted by eight bits.

In a variant, a block is considered not to be constituted by a single binary word but, for example, by several consecutive binary words in the sequence of the message. A first block may thus be constituted for example by the first hundred binary words, that is, the first hundred words of the message, the integrity of which is to be checked, which words each contain a given number of bits, for example eight or sixteen, or any other value compatible with the working formats of the processing circuits of the portable object, the second block then being constituted by the next hundred binary words, and so on until the end of the message.

Of course, it can happen that the total number of words constituting the message does not allow a last block to be constituted with as many binary words belonging to the message as the preceding blocks. This is the case when the quotient of the total number of words constituting the message and the number of words chosen to constitute each block is not a whole number. In such a case the last block cannot contain a number of words coming from the message that is equal to the number contained in the preceding blocks.

This is the case, for example, when a message contains one thousand and thirty words and each block is constituted by one hundred words. Ten blocks of a hundred words can therefore be constituted and only thirty words remain to constitute the last block. In such a case the last block is constituted using these thirty remaining words to which, for example seventy words of zero binary value, that is, containing exclusively "0"s, are added.

To avoid this artifice, one can act such that the number of words constituting each block is an exact divisor of the number of words constituting the message, so that all the blocks will contain only words coming from the original message. This solution has the disadvantage that the processing circuits count the total number of words in the message before constituting the blocks, in order to determine the number of words that have to constitute each block. The processing circuits therefore have to know in advance the optimum number of words in each block, and if this optimum number does not consist of a divisor of the total number of words constituting the message, the processing circuits have to determine what divisor of a lower value should be employed. It will be seen therefore that this solution is rather more cumbersome and difficult to implement; it is however capable of being envisaged.

Figure 5 makes it possible to illustrate how to understand how the blocks are distributed within the message, and how they are interlaced by the processing circuits calculating the first signatures, or the processing circuits of the portable object when they execute the checking calculations.

Figure 5 shows a set m of blocks numbered from b_1 to b_m . As described above, each block can be constituted by a single binary word, or by several binary words associated with one another. By associating a given number of blocks with each other, a module is produced that will subsequently serve as a basis for calculating a signature, as described above.

If the message P is divided into m blocks numbered from B_1 to B_m , it is possible to constitute m/n modules, each of n blocks, such that the module M_i ranking i is constituted by the blocks ranking i , $i + n$, $i + 2n$, and so on as far as $i + m$ with $1 \leq i \leq n$ and $0 \leq r \leq m/n - 1$.

Thus the first module will be constituted by the blocks B_1 , $B_1 + n$, $B_1 + 2n$, ..., $B_1 + rn$ one after another.

The number n must be determined so that the times for calculating and checking signatures are not too long on the one hand, and on the other hand so that, where possible, each module contains exclusively information coming from the original message, or the message to be checked.

If n is not an exact divisor of the number of blocks, certain modules could contain information other than the information relating to the messages on which the signatures have to be calculated. Indeed, certain modules would have to be supplemented, for example with binary "0"s or "1"s.

This is why the number n of blocks constituting each module is preferably chosen such that n is a divisor of the total number of blocks constituting the message.

Irrespective of the number of words constituting each block, or the number of blocks constituting each module, it will be seen that the overlapping of the blocks in the modules and that the signatures are totally independent of the actual structure of the message, which is particularly important where the message is a piece of software that is to be checked for integrity. Indeed, a change made to part of the message because of this overlapping may appear across several signatures, such that the

probability of detecting modifications is increased since several modules might be altered from their original state. This means that the modifications are, to a certain extent, diffused through all or a significant part of the modules. This arrangement therefore allows the number of signature-checking calculations to be reduced in comparison with the arrangement in which each module was considered to be constituted by a certain number of blocks succeeding each other in the whole message.

Another way of constituting the modules, illustrated in Figure 6, consists, rather as before, in dividing the whole message into blocks B1, B2, B3, ... Bm, each having a given number of bits or words. The block B1 is for example constituted by the first k binary words of the message, the block B2 is constituted by the next k words, and so on until the end. There too the number k is chosen, for example, so as to be a divisor of the number of words in the message, such that the last word constituted at the time of the arrangement with a view to calculating the signatures, before memorisation in the portable object, is constituted exclusively of words belonging to the original message, and that there is no artifice requiring the block to be completed with meaningless information.

Each block therefore comprises a given number of bits, each of these bits being locatable by its rank within the block, and a module is constituted by associating one or more bits of a given rank in a block with the bit(s) of the same rank in the other blocks, then the signatures can be calculated by using the modules constituted in this way.

Thus, assuming that a bit is taken from each block in order to constitute the modules, a first module would be constituted by the first bit of the first block, the first bit of the second block, and so on up to the first bit of the last block constituting the message. A second module would be constituted by the second bit of the first block, the second bit of the second block, and so on.

Each module is thus constituted by a chain of information across the message in question, and it then becomes very difficult for a consistent modification of part of

the message to go unnoticed when the signatures are recalculated, whatever the number of signatures recalculated and compared with the initial number of signatures. Indeed, the instructions in the case of software, or the data, are generally written in succession longitudinally, and the slightest consistent modification of the program or data would result in a different transverse chain that would be almost impossible to handle simultaneously. Furthermore, this solution allows the number of signatures that have to be recalculated and compared with the corresponding original signatures, when the message is checked, to be reduced even further.

Rather than choosing a transverse chain, as defined above, a module can be constituted using a pseudo-transverse chain of information that takes a given number of bits at random in each block, when each signature is calculated before memorisation. Thus the first bit of the first block can be associated with the last bit of the second block, then associated with a bit of a different rank in the third, and so on. Of course, such an association requires the use of a random reference number, picked when the signature is calculated before memorisation, for example. This random reference number is used by the processing circuits to determine the series of bits to be considered, and must be memorised in the memory circuits of the portable object, such that, at the time of checking, the processing circuits know how they have to perform the distribution again in order to constitute the modules from the message that is to be checked.

Furthermore, any other type of variant for interlacing blocks in order to constitute modules, or for interlacing words or even bits to make up blocks, can be envisaged. In particular, rather than following a logical sequence when interlacing words or blocks, it may be envisaged that the interlacing of words to make up a block or blocks, in order to constitute a module, can take place at random. To this effect, for example, before the processing circuits constitute the modules with a view to calculating their signatures for memorisation, they will have to determine how the modules are reconstituted, and the parameters used for constituting the modules will have to be memorised so that subsequent checks can be conducted. Thus, it may be considered that the processing circuits prepare a series of n random numbers,

n corresponding to the number of blocks that will constitute each module, the message containing a number m of modules, and the series of random numbers $V_1, V_2, V_2, \dots V_n$ makes it possible to determine, from a block that will constitute the first block of a module, which other blocks will constitute this module.

In this case it is also necessary to keep a record of the series of random numbers that have been used so that the modules can be reconstituted on the basis of the message that has to be checked.

Although the use of a secret key is preferable, in order to make the check fail-safe, there are however cases when this is not necessary, whether or not the message is divided into modules and serves for the preparation of one or more distinct signatures. It simply gives greater security, since it prevents two different portable objects, used for calculating signatures for the same message, from containing identical signatures, since the secret key contained in each of the portable objects is different. This limits the risk of fraud in the case where a person intending to commit fraud observes what is happening with a message such as a piece of software, and attempts to deceive the signature-checking means. The use of a secret key is specially important when the software has to be transferred from one person to another for example. However, when the calculation of the signature(s) of a message is required by an end user, with a view to memorising them, for subsequent checking, it is not necessary for the portable object used for checking to contain a secret key. The signature of the message can be obtained by simply converting the data it contains, taken as a whole or in distinct modules, using an algorithm, so that each signature is a simple image of the data used in order to calculate it. Indeed, where it is an end user who wants to be able to check the integrity of his data storage medium, as and when he uses it, he has no interest in seeking to deceive the system. In such a case, a signature can result from a simple compression of information.

The calculation of a signature related to a message or part of a message would be a function only of that message or that part of the message subjected to the algorithm in question.

A portable object for calculating signatures with a view to checking them will therefore not, as is the case in Figure 3, comprise a memory area 10 incorporating the secret key K, but on the other hand it could contain one or more memory areas 10 each containing a signature S1, S2, Sm, in the same way that it would comprise a processor with processing circuits 11 for implementing an algorithm A.

Accordingly, the generation of a signature as described in relation to Figure 2, will no longer involve the use of the secret key K at the time of each intermediate operation required by calculating the signature.

A system for implementing the invention comprises a portable object 1 comprising at least one memory area 10 for memorising at least one software signature, and processing circuits 11 memorising an algorithm 12, in order at least to perform the operation of recalculating the signatures after at least one original signature has been recorded. The fact that the recalculation takes place inside the processing circuits of the portable object makes it possible to prevent an outside observer from seeing the value of the recalculated signature, even if this observer knew the signature initially calculated, with a view to introducing it into the memory of the portable object, when said memorised signature had been calculated by processing circuits outside the portable object, to avoid having very lengthy calculation times, for example.

The processing circuits of the portable object therefore contain an algorithm or encoding program A, in order at least to convert a message M, the integrity of which has to be checked, and the memory can also contain a key K where even greater security is to be conferred. Furthermore, where the memory of the portable object contains a large number of signatures each belonging to a different module of an initial message, the processing circuits of the portable object should be arranged so that they can perform the check on a number of modules less than the initial number, to reduce calculation times. The processing circuits must also be in a position to reconstitute the modules of the message to be checked, in the same way that they were constituted when the signatures were calculated and memorised.

Of course, the memory area in which the signatures are memorised is a non-volatile memory area.

Furthermore, as indicated above, the portable object 1 cannot function alone, and needs to be associated with other means 2 for constituting a system for implementing the invention. In particular, interface circuits must be constituted between the portable object and the computer, or the processing device for the software or data, the integrity of which is to be checked. It is via this processing device or this computer that the information to be checked, and the information constituting the original message, are handled in the portable object, following a dialogue established between the processing circuits of the portable object and the processing circuits of the associated processing device. Among other things, the keyboard generally associated with processing devices, or other data-entry means (mouse, touch screen, etc.) can be used for establishing the dialogue with the portable object, especially when there is a question of entering confidential access keys, or identifiers corresponding to the message, the integrity of which should be checked against the original messages.

The interface circuits can be directly incorporated into the processing device or the computer, or they can be located outside, and connected via a link. Of course, a connector is provided between the portable object and the interface or coupling circuits.

When the system is used for checking, in the case where a single signature related to the whole of the original message has been memorised, the processing circuits of the portable object take the whole message, the integrity of which is to be checked, and calculate the signature of this message to be checked, by implementing the algorithm that they contain, then check whether or not the stored signature matches the recalculated signature. This applies to short messages.

In contrast, when the initial message has been divided into several modules, because it is relatively long, and consequently several signatures have been memorised in the memory of the portable object, then the processing circuits of the

latter will determine, in so far as it has not already been determined, a number p of modules that have to be signature-checked, and also their serial numbers. They then reconstitute the modules to be signature-checked on the basis of their serial numbers, in the same way as the modules that are supposed to correspond in the original message had been constituted when all the signatures were precalculated.

Preferably, in order to prevent a person intending to commit fraud from finding out the part of the message for which signatures have been calculated for checking, the whole message is sampled by the processing circuits of the portable object, and it is only inside the latter that sorting takes place. Of course, since the message can be very long, significantly greater than the memory capacity of the portable object, the processing circuits can confine themselves to reading the data constituting the message to be checked, as they pass, and using only the data that may serve as a basis for the signatures to be checked.

Moreover, as has been mentioned before, any one card can contain the signatures associated with several pieces of software. To be able to distinguish these signatures, a control area can be provided in the portable object, containing information as to the identity of each piece of software for which at least one signature has been memorised, this control area also indicating to the processing circuits the memory addresses where the signatures relating to a given original message are located. This can be a serial number, or any other type of information making it possible to identify which message has to be subjected to a comparison. In such a case, at the time of checking, provision is made for the system to ask the user for the number or identity of the message to be checked.

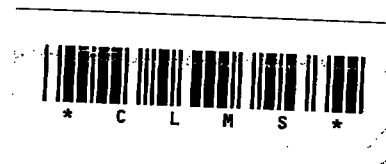
Finally, in an improved variant it is provided that, where the message is intentionally modified for updating purposes, each corresponding signature memorised in the portable object can be updated. In this case, under the control of the user, a complete re-recording of the new signatures corresponding to the modified message can take place, either in another memory area of the portable object or in the same area, which is then for example of the EEPROM type, that is, it is electrically erasable and reprogrammable under the control of the circuits of the portable object.

This is fully within the scope of the person skilled in the art, since portable objects of the memory-card type with electronic microcircuits generally incorporate memories of this type and a terminal can be provided as necessary for programming or erasing certain memory areas, at the level of the terminals connecting these portable objects with an external system, in addition to the terminals needed for power supply and data transfer. In other cases the programming voltages are supplied by the portable object itself. However, in each case, the erasing and re-recording of new signatures in the memory take place selectively, and affect only the areas to be modified.

The invention is therefore particularly advantageous, since it allows the integrity of a message constituted by a piece of software and/or data memorised on a computer storage medium to be ensured in a simple, safe and relatively inexpensive way. This can involve messages that have been loaded from an original and have been modified, either following an intrusion into the actual site where they are located, or remotely via a transmission line.

Naturally, modifications can be made to the process and the system for implementing it without thereby leaving the scope of the present invention, as defined by the claims.

CLAIMS



1. A process for checking the integrity, against an original message, of a subsequent message containing information, consisting of causing processing means to calculate a signature from the original message (M) to which an algorithm (A) is applied, characterised in that it comprises the steps consisting of:
 - dividing the original message into a total number (m) of modules (M1, M2, ... Mm);
 - using said processing means to apply said algorithm (A) to each module of the original message to calculate a respective signature (S1, S2, ... Sm) of each module;
 - using an electronic portable object having processing circuits (11) and memorising means including a protected non-volatile memory area (10) accessible only by said processing circuits, and memorising said signatures in the protected memory area and said algorithm in the memorising means;and checking the subsequent messages in the portable object by the steps consisting of:
 - selecting, in the subsequent messages, a partial number (p) of modules less than the total number (m) of modules, said partial number being such that the probability of detecting an alteration in the subsequent message, by considering only said partial number of modules, has a given value;
 - using the processing circuits to apply said algorithm to each selected module so as to calculate a respective signature;
 - selecting, in the protected memory area, the signatures of the modules of the original message that correspond to the selected modules of the subsequent message and comparing these signatures with those of the subsequent message; and
 - causing the portable object to send a response indicating whether the integrity between the original and subsequent messages is confirmed, i.e. whether said signatures of the original message correspond respectively to those of the subsequent message.



2. A process according to Claim 1, characterised in that the modules are located by a number and, to determine the number of modules on which the signature check will be made, the processing circuits of the portable object make (p) successive choices of different random numbers, each random number determining the number of a module to be checked.
3. A process according to Claim 2, characterised in that, to determine the number of modules on which the signature check must be made, from the (m) modules contained in the original message, the processing circuits of the portable object choose a binary number (a) of length (m) expressed in bits such that the length of the random number chosen is directly representative of the number of modules contained in the original message, and in that the value of the random number comes from a code (p) out of (m), i.e. of the (m) bits contained in the random number, there are (p) that have a certain binary value (1 or 0) while the remaining (m - p) bits have the complementary value, and that each bit of the random number can be located by a different serial number; the number of modules to be signature-checked is determined by the serial number of the (p) bits taken from the (m) bits in the random number.
4. A process according to Claim 1, characterised in that a message is constituted by a sequence of bits locatable by their serial numbers or addresses as a function of the positions they occupy in the message, and that the processing circuits create the different modules (M1, M2, ... Mm) from the original message in order to calculate the signature of each of these modules before bringing about its memorising by the processing circuits of the portable object, as follows:
 - each module is constituted by taking a certain number of message bits according to a predetermined rule and/or a rule created by the processing circuits of the portable object, taking into account random elements for example, and the rule, having served to constitute each module from the original message, is kept in the portable object so as to reconstitute, from a message to be checked, the modules of this message according to the same rule used for constituting the modules from the original message.



5. A process according to Claim 1, characterised in that the message is organised in the form of binary words, each containing a given number of bits, a piece of information constituting a software instruction or part of an instruction, or a datum or part of a datum, the integrity of which must be preserved, for the correct running of a program using these instructions and/or data, the constituting of modules (M1, M2, ... Mm) with a view to calculating the signature of at least some of them, with a view to checking the integrity of the message, consists of causing the processing circuits that have to calculate the signatures considered, first to constitute a certain number (n) of blocks (B1, B2, ... Bn) each containing a certain number of pieces of information such that each piece of information or part of a piece contained in a block can be located by the positions of the bits that constitute this piece of information or part of a piece in the block considered, and in that it consists, for constituting a given module, of causing the processing circuits to take at least one bit from each of the blocks according to a given rule.
6. A process according to Claim 5, characterised in that the grouping of the information in order to form blocks is carried out by taking the information in the order in which it appears when the message is read.
7. A device for checking the integrity, against an original message, of a subsequent message containing information, arranged to cause processing means to calculate a signature (S) from the original message (M) to which an algorithm (A) is applied, characterised in that it comprises a portable object comprising means for connection to a device (2), such as a computer, able to execute the message, said portable object also comprising processing circuits (11) and memorising means including a protected non-volatile memory area (10) accessible only by the processing circuits, in which:-
 - the original message (M) is divided into a total number (m) of modules (M1, M2, ... Mm) and the protected memory area memorises a plurality of signatures (S1, S2, ... Sm), each signature being the result of applying the algorithm (A) to a given module of the original message;the portable object containing the algorithm (A) and being arranged so as to:



- select, in the subsequent message, a partial number (p) of modules less than the total number (m) of modules, said partial number being such that the probability of detecting an alteration in the subsequent message, by considering only the partial number of modules, has a given value;
- use the processing circuits to apply said algorithm to each selected module so as to calculate a respective signature;
- select, in the protected memory area, the signatures of the modules of the original message that correspond to the selected modules of the subsequent message and compare these signatures with those of the subsequent message; and
- cause the portable object to send a response indicating whether the integrity between the original and subsequent messages is confirmed, i.e. whether said signatures of the original message correspond respectively to those of the subsequent message.