

Method for the execution of a secure software application and a non-secure software application, which are interlaced one with the other

The invention relates to the field of methods for the interlaced execution of secure software and non-secure software applications, at least one output datum generated at the output of a module of said non-secure software application being used as input datum applied at the input of a module of the secure software application.

A software application is a computer program whereof the instructions, stored in a memory of a computer, can be executed by a processor of the computer.

A secure software application is a software application that must be executed securely, i.e. so as to guarantee proper operation and with a predetermined trust level. This means that, during the execution of that secure software application, the occurrence of a defect in contradiction with a security requirement has a predetermined likelihood. For example, standard IEC 61 508 defines several security levels from levels SIL0, high likelihood of occurrence of a defect, to SIL4, low likelihood of occurrence of a defect. For a secure execution, the computer executing the instructions of the secure software application must meet certain operating criteria. This is referred to as a secure computer.

In the present document, the term "non-secure software" refers to a software application whereof the execution is not constrained in terms of reliability or is less constrained than the secure software application. For example, if the secure software application is a level SIL2, a software application with level SIL0 will be considered a non-secure software application. In the present document, non-secure software applications are executed on a computer called "non-secure computer", i.e. a computer meeting operating criteria compatible with the security level of the non-secure software application.

A software application is made of a plurality of elementary modules.

The execution by the processor of a computer of the instructions corresponding to a module of the software application constitutes an elementary process or task.

A module includes an input section for acquiring input data, for example by reading current values of the input data in a predetermined memory space of the computer; a processing section for computing output data values from values acquired from the input data; and an output section for writing computed values of the output data in a predetermined memory space of the computer.

Except in case of dysfunction, once a module is launched, its execution is always done without interruption until it is completed, i.e. until output data have been generated.

Based on the architecture of the software application, the output data of the first module may constitute the input data of a second module. A data stream therefore exists

beginning with an initial event, such as an initial datum updated at the input of the software application, and ending with an output event, such as the end of the execution of a module, passing through different intermediate data exchanged between the modules of the software application.

5 A complete data stream covers several successive execution cycles of the software application. In fact, upon each cycle, the only modules that are actually executed are those for which updated input data are available at the beginning of the cycle or will be available during the cycle when the considered module is executed. A module will be executed only once on an updated input data.

10 Two software applications are said to be interlaced when a module of one of the two software applications admits, among its different input data, at least one output data from a module of the other software application. One thus has a data stream in which there is an exchange of data between the two executed software applications, such that the two software applications are interdependent.

15 Figure 1 shows an example of a logic structure including two interlaced software applications: a first secure software application L_S and a second non-secure software application L_NS. This illustrative example will be used throughout the description.

 The logic structure includes a first module M1, belonging to the first software application L_S. The first module M1 uses an updated initial datum D0 as input. This is for
20 example a datum delivered by a sensor and updated regularly. The first module M1 is capable of generating a first output datum D1. Thus, the data stream begins with the initial event consisting of executing module M1.

 The logic structure includes a second module M2, belonging to the second software application L_NS. The second module M2 uses the datum D1 generated at the
25 output of the module M1 as input data. The module M2 is capable of generating a second output datum D2.

 The logic structure includes a third module M3 belonging to the first software application L_S. The third module M3 uses the datum D2 generated at the output of the
30 module M2 as input datum. The module M3 is capable of generating a third output datum D3.

 The logic structure includes a fourth module M4, belonging to the second software application L_NS. The fourth module M4 uses the datum D3 generated at the output of the module M3 as input datum. The module M4 is capable of generating a fourth output datum D4.

35 The logic structure lastly includes a fifth module M5 belonging to the first software application L_S. The fifth module M5 uses the datum D4 generated at the output of the

module M4 as input datum. The fifth module M5 does not generate any output datum. The data stream ends with the final event consisting of the end of execution of module M5.

The first software application L_S being a secure software application, the instructions of the first, third and fifth modules, M1, M3 and M5, must be executed
5 securely by a secure computer.

Two methods are known to execute this logic structure, where a secure software application and a non-secure software application are interlaced.

According to a first known execution method, the secure L_S and non-secure L_NS software applications are integrated into a common secure software application that
10 is executed, securely, on a secure computer C1.

Figure 2 diagrammatically shows the first known execution method for the case of the logic structure of figure 1.

During an execution cycle N, the common secure software is executed.

It begins by executing the first module M1. The first module M1 is in fact executed
15 when the initial datum D0 has been updated. The first module M1 leads to the generation of a first output datum D1.

Then, still during the cycle N, the execution of the common secure software application continues through the execution of the second module M2. This module is actually executed because it has the required input datum, in this case the datum D1 that
20 has just been generated by module M1. The module M2 leads to the generation of the second output datum D2.

Still during the cycle N, the execution of the common secure software application continues through the execution of the third module M3. This module is actually executed because it has the required input datum, i.e. datum D2 that has just been generated by
25 the second module M2. The module M3 leads to the generation of a third output datum D3.

Still during the cycle N, the execution of the common secure software application continues through the execution of the fourth module M4. This module is actually executed because it has the required input datum, i.e. the datum D3 that has just been
30 generated by the third module M3. The module M4 leads to the generation of a fourth output datum D4.

Lastly, still during the cycle N, the execution of the common secure software application continues through the execution of the fifth module M5. The latter is actually executed because it has the required input datum, i.e. the datum D4 that has just been
35 generated by the module M4. The end of the execution of the fifth module M5 ends the

execution of the common secure software application. The latter has been completely executed during a single cycle N.

Thus, according to this first known execution method, the non-secure software application L_NS is in fact executed securely by a secure computer.

5 Taking the security constraints into account for the execution of the different modules of the non-secure software application L_NS imposes an overwork during the development of the non-secure software application L_NS so as to integrate it with the secure software application L_S, into the common secure software application.

10 Furthermore, during the execution of the common secure software application, the time needed to execute the modules of the non-secure software application L_NS is greater than when those modules are executed on a non-secure computer.

However, the complete execution of the common secure software application is done during a single execution cycle, leading, despite everything, to good overall time performances.

15 According to a second known execution method, the secure L_S and non-secure L_NS software applications are executed in parallel, on two separate computers: the first secure software application L_S is executed by a first secure computer C1, while the second non-secure software application L_NS is executed by a second non-secure computer C2.

20 Figure 3 diagrammatically shows the second known execution method for the case of the logic structure of figure 1. The execution cycles for the software applications on the first and second computers, respectively, are substantially synchronous. Only one data stream is shown.

25 During the first execution cycle N of the secure software application L_S on the first computer C1, the first module M1 is executed on an updated initial datum D0, so as to generate the first output datum D1.

The modules M3 and M5 of the secure software application L_S, which do not have updated data D2 and D4, are not executed.

30 In parallel, during a first execution cycle N of the non-secure software application L_NS on the second computer C2, the modules M2 and M4, which do not have updated data D1 and D3, are not executed.

During a second execution cycle N+1, on the first computer C1, the module M1 already having been executed, it is not executed again.

35 The modules M3 and M5 of the secure software application L_S, which do not have updated input data D2 and D4, are not executed.

In parallel, on the second computer C2, the module M2 of the second software application L_NS, which now has an updated input datum D1, is executed. It generates the second output datum D2.

However, the module M4, which does not have the updated input datum D3, is not executed.

During a third execution cycle, N+2, on the first computer C1, the module M1 is not executed again.

The third module M3 of the first software application L_S, which now has an updated input datum D2, is executed. It generates the third output datum D3.

The module M5, which does not have the updated input datum D4, is not executed.

In parallel, on the second computer C2, the module M2, which has already been executed, is not executed again.

The module M4, which does not have the updated input datum D3, is not executed.

During a fourth execution cycle, N+3, on the first computer C1, the modules M1 and M3, which have already been executed, are not executed again.

The module M5, which does not have the updated input datum D4, is not executed.

In parallel, on the second computer C2, the module M2, which has already been executed, is not executed again.

However, the module M4, which now has the updated input datum D3, is executed. It delivers the datum D4 as output.

Lastly, during a fifth execution cycle, N+4, on the first computer C1, the modules M1 and M3, which have already been executed, are not executed again.

The module M5, which has the updated input datum D4, is executed.

In parallel, on the second computer C2, during the cycle N+4, the modules M2 and M4, which have already been executed, are not executed again.

The data stream initiated in cycle N by the update of the initial datum and the execution of the first module M1 ends with the cycle N+4 with the end of the execution of the fifth module M5.

This known execution method is simple to implement and allows secure execution of only the secure software application L_S. The development of the non-secure software application L_NS is not constrained by an execution in a secure environment, contrary to the first known execution method.

However, since a module of a software application must wait that an updated input data is available, this second known execution method requires a large number of execution cycles. Consequently, the overall time performance of this second method is poor.

5 The invention therefore aims at solving the aforementioned problems.

To that end, the invention relates to a method for the executing of interlaced secure and non-secure software applications, at least one output datum generated at the output of a module of a non-secure software application being used as input datum applied at the input of a module of the secure software application, characterized in that it consists of:

- 10 a) – executing, with the security constraints corresponding to those of the non-secure software application, said secure and non-secure software applications grouped together in a common non-secure software application, so as to update said at least one output datum of said module of the non-secure software application; and
- 15 c) – executing the secure software application, with security constraints corresponding to those of the secure software application, by using said at least one updated output datum as input datum for said module of the secure software application.

According to specific embodiments, the method includes one or more of the following steps, considered alone or according to any technically possible combinations:

- 20 - between steps a) and c), the method includes a step b) consisting of transferring, from said non-secure common software application to said secure software application, said at least one updated output datum.

- steps a) and c) are carried out successively over time.
- step a) is implemented by a second non-secure computer and step c) is implemented by a first secure computer, the first and second computers being physically segregated from one another.

- 25 - steps a) and c) are implemented by a single computer capable of guaranteeing segregation between the non-secure execution of the common non-secure software application in step a) and the secure execution of the secure software application in step c).

- 30 - steps a) and c) are carried out parallel in time, the secure execution of one module of the secure software application being stopped as long as an input datum of said module of the secure software application that corresponds to an output datum of a module of the non-secure software application has not been updated.

- step a) is implemented by a second non-secure computer and step c) is implemented by a first secure computer, the first and second computers being physically segregated from one another.

The invention and its advantages will be better understood upon reading the following description, provided solely as an example, and done in reference to the appended drawings, in which:

- figure 1 is a diagrammatic illustration of the logic structure of an interlaced secure software application and non-secure software application;

- figure 2 is a diagrammatic illustration of a first known method for executing the logic structure of figure 1;

- figure 3 is a diagrammatic illustration of a second known method for executing the logic structure of figure 1;

- figure 4 is a diagrammatic illustration of a device for implementing the execution method according to the invention;

- figure 5 is a diagrammatic illustration of a method according to the invention for executing the logic structure of figure 1, implemented on two computers, like those of the device of figure 4;

- figure 6 is a diagrammatic illustration of the method of figure 5, where several data streams are shown; and

- figure 7 is a diagrammatic illustration of a method for executing the logic structure of figure 1, implemented by a single computer.

In a first embodiment shown in figure 4, the method is implemented on a device including two computers.

The device 10 includes a hardware layer 12 and a software layer 32.

The hardware layer 12 includes:

- a first computer C1, including a first processor 16 and a first memory 18;

- a second computer C2, including a second processor 26 and a second memory

28.

The first computer C1 is a secure computer, while the second computer C2 is a non-secure computer.

The first and second computers, C1 and C2, are connected to each other by a suitable communication link 30 allowing the exchange of data. This is for example an asynchronous or synchronous serial link.

The software layer 32 includes a software application 34 that corresponds to the secure software application L_S for the first computer C1, and a common non-secure

software application 36 grouping together the interlaced secure software application L_S and non-secure software application L_NS for the second computer C2.

The software layer 32 also includes a transmission software application 38 for the first computer C1, and a transmission software application 40 for the second computer C2. The transmission software applications 38 and 40 are capable of transmitting and receiving messages suitable for transferring the value of a datum stored in the memory of the second computer C2 to the memory of the first computer C1, and vice versa.

The software applications 38 and 40 allow segregation of the first and second computers C1 and C2.

The transmission software application 38 participating in the secure environment within which the secure software application L_S is executed must be executed securely.

The execution method for a data stream is shown diagrammatically in figure 5.

A first execution cycle N begins with the update of the initial datum D0.

In a first step 100, the common non-secure software 36 is executed on the second computer C2. It is executed freely, or at the very least with the security level associated with the non-secure software application L_NS.

The execution of the common non-secure software application 36 begins with the execution of the first module M1. The latter is in fact executed because it has an updated input datum, in this case the datum D0. The first module M1 leads to the generation of a first output datum D1.

Then, still during the same cycle N, on the second computer C2, the second module M2 is executed, since it has the required updated input datum via the datum D1 that has just been determined. The execution of the second module M2 leads to the generation of a second output datum D2.

Still during the same cycle N, on the second computer C2, the third module M3 is executed. Indeed, it has the required updated input datum via the datum D2 that has just been determined. The execution of the third module M3 leads to the generation of a third output datum D3.

Still during the cycle N, on the second computer C2, the fourth module M4 is executed, since it has the required updated input datum via the datum D3 that has just been determined. The execution of the fourth module M4 leads to the generation of a fourth output datum D4.

Lastly, still during the cycle N, on the second computer C2, the fifth module M5 is executed, since it has the required input datum via the datum D4 that has just been determined. The end of execution of the fifth module M5 corresponds to the end of execution of the non-secure common software application 36 and step 100.

The first cycle N thus makes it possible to execute the entire common non-secure software application 36. The first cycle N in fact corresponds to an execution of the common non-secure software application 36 on the non-secure computer C2 for the considered data stream.

5 Then, during a second step 110, the software applications 38 and 40 are executed. They make it possible to transfer the values of the output data from the modules of the non-secure software application L_NS, which constitute the input data for the secure software application L_S, from the memory of the second computer C2 using the link 30. The transmission execution of the software application 40 corresponds to the end of the
10 first cycle N, while the reception execution of the software application 38 corresponds to the beginning of the following, second cycle N+1.

The second cycle N+1 continues via a third step 120, during which the software application 34, which corresponds only to the secure software application L_S, is executed securely on the first computer C1.

15 When a module of the secure software application L_S needs the value of an input datum corresponding to an output datum of a module of the non-secure software application L_NS, it reads the value of that input datum in the memory of the first computer C1. This value has been generated by the second computer C2, during the first step 100 of the method, and transferred into the memory of the first computer C1 during
20 the second step 110.

The execution of the secure software application L_S begins by the execution of the first module M1. This first module M1 is executed on the initial datum D0 updated at the beginning of the preceding cycle N. The first module M1 leads to the generation of a first output datum D1.

25 Then, still during the second execution cycle N+1, the third module M3 is executed, since it has an updated input datum D2. This updated datum corresponds to the datum D2 determined by the execution of the second module M2 of the non-secure software application 36 during the first step 100. The third module M3 leads to the generation of a third output datum D3.

30 Then, still during the second execution cycle N+1, the fifth module M5 is executed, since it has an updated input datum D4. This updated datum corresponds to the datum D4 determined by the execution of the fourth module M4 of the non-secure software application 36 during the first step 100. The end of the execution of the fifth module M5 corresponds to the end of the execution of the secure software application L_S and step
35 120.

The second cycle N+1 makes it possible to execute the entire software application 34. The second cycle N+1 in fact corresponds to an execution of the secure software application on the secure computer C1 for the considered data stream.

5 The common non-secure software application 36, grouping together the secure L_S and non-secure L_NS software applications, which is not designed to be executed securely, is simpler to develop and hone. In fact, it requires taking the security constraints associated with the security level of the non-secure software application L_NS into account.

10 A single first cycle is necessary for the execution of the common non-secure software application 36. The duration of this first execution cycle is generally shorter than that of the common secure software application according to the first known execution method from the prior art, which requires a secure execution. For there to in fact be a gain in execution time, it is of course necessary for the additional non-secure execution time of
15 the modules of the secure software application L_S in the common non-secure software application according to the first method to be shorter than the additional time due to the secure execution of the modules of the non-secure software application L_NS in the common secure software application according to the method of the prior art.

20 Preferably, during the first step of the method, only the modules of the secure software application L_S that are necessary to generate input data for the modules of the non-secure software application L_NS are in fact executed. To that end, the common secure software application 36 advantageously integrates a call means. Each time a module of the non-secure software application L_NS needs an input datum that corresponds to an output datum of a module of the secure software application L_NS, the
25 call means launches the non-secure execution of the corresponding module so as to generate the required datum. Thus, for example, to reduce the duration of the first cycle, it is not necessary to execute the fifth module M5.

30 In the third step 120, during the second cycle N+1, none of the modules of the non-secure software application L_NS are executed. Only the modules of the secure software application L_S are in fact executed.

35 In the example described here, steps 100, 110 and 120 were carried out successively over time, the end of the execution of the common non-secure software application 36 triggering that of the software applications 38 and 40, and the end of the execution of the software applications 38 and 40 triggering that of the software application 34. However, alternatively, the two executions on the two computers may begin simultaneously. The execution of the software application 34, i.e., the secure software

application L_S, which is on standby until it has an updated value for the input datum D2 to execute the third module M3, then again putting itself on standby until it has an updated value for it the input datum D4 for the execution of the fifth module M5. In this alternative, the exchange of data between the two computers is done as the data are produced. The transmission means 38 and 40 are adapted accordingly.

Figure 6 reiterates figure 5, but makes it possible to follow several data streams. A first stream initiated in cycle N by an update of the initial datum D0 and the execution of the first module M1 ends in cycle N+1 through the end of the execution of the module M5.

A second stream initiated in cycle N+1 by an update of the initial datum D0 and the execution of the first module M1 ends in cycle N+2 through the end of the execution of the module M5.

In still another alternative shown in figure 7, the method is implemented on a device that includes a single computer C3 capable of constituting a secure environment for the secure execution of a secure software application and constituting a non-secure environment for the non-secure execution of a non-secure software application. The computer C3 is thus capable of segregating the secure executions of the software application 34 (secure software application L_S) and non-secure executions of the common non-secure software application 36 (non-secure L_NS and secure L_S software applications).

In this alternative, steps 100, 110 and 120 of the method are carried out successively over the course of a single execution cycle N of the common non-secure software application 36, then the software application 34 (i.e., the secure software application L_S).

More generally, the method according to the invention may be implemented by different devices.

12
CLAIMS

1.- A method for the execution of a secure software application (L_S) and a non-secure software application (L_NS), said application being interlaced one with the other,
5 at least one output datum generated at the output of a module of said non-secure software application being used as input datum applied at the input of a module of the secure software application, characterized in that it consists of:

a) – executing (100), with security constraints corresponding to the security constraints of the non-secure software application (L_NS), said secure (L_S) and non-secure (L_NS)
10 software applications grouped together in a common non-secure software application (36), so as to update said at least one output datum of said module of the non-secure software application; and,

c) – executing (120) the secure software application (L_S), with security constraints corresponding to the security constraints of the secure software application (L_S), by
15 using said at least one updated output datum as input datum for said module of the secure software application (L_S).

2.- The method according to claim 1, wherein between steps a) and c), the method includes a step b) consisting of transferring (110), from said non-secure common
20 software application (36) to said secure software application (L_S), said at least one updated output datum.

3.- The method according to claim 1 or claim 2, wherein steps a) and c) are carried out successively over time.

4.- The method according to any one of claims 1 to 3, wherein step a) is implemented on a second non-secure computer (C2) and step c) is implemented on a first secure computer (C1), the first and second computers being physically segregated from one another.

5.- The method according to any one of claims 1 to 3, wherein steps a) and c) are implemented on a single computer (C3) capable of guaranteeing segregation between the non-secure execution of the common non-secure software application in step a) and the secure execution of the secure software application in step c).

6.- The method according to claim 1 or claim 2, wherein steps a) and c) are carried out parallel in time, the secure execution of one module of the secure software (L_S) application being stopped as long as an input datum of said module of the secure software application that corresponds to an output datum of a module of the non-secure software application has not been updated.

7.- The method according to claim 6, wherein step a) is implemented on a second non-secure computer (C2) and step c) is implemented on a first secure computer (C1), the first and second computers being physically segregated from one another.