

Computer Implemented Method for Preventing Rowhammer Attack in a Computing Device

Field

- 5 The present invention relates to a computer implemented method for preventing rowhammer attack in a computing device.

Background

- 10 Rowhammer is an unintended side effect in dynamic random-access memory (DRAM) that causes memory cells to leak their charges and interact electrically between themselves, thereby possibly altering contents of nearby memory rows that were not addressed in the original memory access. This vulnerability may be triggered by specially crafted memory access patterns that rapidly activate the same memory rows numerous times.
- 15 Consequently, due to the repeated access of a certain memory cell, nearby memory cells' content may be changed, resulting in memory bit flip (i.e. bit value 0 becomes bit value 1, or vice-versa).

- 20 So, rowhammer attacks are exploits that can modify or cause bit flip in unreachable physical memory by accessing nearby reachable physical memory. That is, rowhammer attacks are advanced type of data attacks. Due to design of physical memory, an attacker usually accesses an aggressor row (of the physical memory) in a certain pattern to intentionally affect a victim row (of the physical memory), in order to cause bit flip in the victim row. As shown in FIG.
- 25 1, the victim row's memory cell may be flipped from bit value 1 to bit value 0, based on the crafted attack pattern shown in the pseudo code.

A typical rowhammer attack comprises the following steps:

1. Rowhammer Testing: scan memory for useful bit flips;
- 30 2. Memory Manipulation: locate a crucial data structure, and map the data structure to the flips; and
3. Data Oriented Programming (DOP) attack: modify the located data structure and execute an intended payload.

Rowhammer attacks are prevalent on both desktop systems and mobile devices, which can undesirably cause privilege escalations, e.g. sandbox bypassing, virtual machine escaping, mobile system rooting and breaking of various system isolations.

5

One way to prevent rowhammer attacks is by using a proposed hardware feature to detect possible harmful memory access patterns and to refresh the physical memory in a fine grained manner. This method however requires adoption of new hardware, meaning it is not directly applicable to all existing systems.

10

Another way is to use software isolation techniques to isolate different memory entities. This may restrain the rowhammer effect to a minimal area of the physical memory, thereby preventing privilege escalation issues. However, this method requires many heavy system modifications and also the resulting memory consumption from running said software techniques may be high.

15

To be practically useful, a rowhammer prevention method should preferably satisfy the following criteria:

1. High security: The rowhammer attack can be prevented.
- 20 2. Low performance impact: Both memory consumption and CPU overhead from running the method should preferably be low.
3. Transparency: No need for modifications to program source code or program binary, i.e. existing system features should not be altered.
4. Application: The method may readily be applied to all existing systems,
25 even without the support of new hardware features.

Two conventional solutions to prevent rowhammer attacks are briefly discussed below for reader's understanding.

30 • **Intel Target Row Refreshing**

Intel™ proposed the Target Row Refreshing (TRR) technique to prevent rowhammer attacks. Internally, TRR identifies possible victim rows, by counting a number of row activations and comparing the counted number against predefined chip-specific maximum activate count (MAC) and maximum activate
35 window (tMAW) values. These rows are then refreshed to prevent bit flips. The

MAC value is a maximum total number of row activations that may be encountered on a particular DRAM row within a time interval that is equal or shorter than the tMAW amount of time before the neighboring rows are identified as victim rows. TRR may also flag a row as a victim row, if the sum of row activations for its two neighboring rows reaches the MAC limit within the tMAW time window. A drawback of TRR is the necessary hardware features to enable TRR are required to be present in processors, which means TRR is not easily deployable to existing systems.

• **B\G-CATT**

B\G-CATT proposed two software-based defensive methods to prevent rowhammer attacks. The first defensive method, B-CATT, extends the system bootloader to disable vulnerable physical memory. B-CATT is highly practical, since it does not require changes to the operating system, and can be deployed on virtually all x86-based systems. While B-CATT is able to stop all known rowhammer attacks, it does not yet tackle the fundamental problem of missing memory isolation in physical memory. So to address the problem, the second defensive method, G-CATT, is proposed which is a generic solution that extends the physical memory allocator of the operating system to physically isolate the memory of different system entities (e.g. kernel and user space). But B\G-CATT methods have high memory consumption. Moreover, B\G-CATT methods do not protect the shared memory, which lowers the security usefulness afforded.

One object of the present invention is therefore to address at least one of the problems of the prior art and/or to provide a choice that is useful in the art.

Summary

According to a 1st aspect of the invention, there is provided a computer implemented method for preventing rowhammer attack in a computing device.

The device has memory which includes a plurality of untrusted memory portions and a plurality of trusted memory portions, the plurality of untrusted memory portions and trusted memory portions being distributed within the memory. The method comprises: determining respective trusted memory portions that are arranged physically contiguous to respective at least one untrusted memory portion to provide an identified set of trusted memory portions, the respective

untrusted memory portions being accessible to perform the attack on the respective trusted memory portions; determining trusted memory portions in the identified set of trusted memory portions that are movable to a different memory location in the memory; moving the movable trusted memory portions to an available memory location in the memory to form a block of trusted memory portions; generating at least one memory guard portion; and configuring the memory guard portion to be arranged at an end of the block of trusted memory portions that is physically contiguous to an untrusted memory portion. The memory guard portion prevents electrical-interaction between the block of trusted memory portions and the untrusted memory portion to prevent the attack.

Preferably, the method may be configured to be incorporated in a memory allocator and a bootloader of an operating system of the device.

Preferably, each trusted memory portion may include being secured against access by unauthorized software processes executed on the device.

Preferably, the memory may include dynamic random-access memory or NAND flash memory.

Preferably, the memory guard portion may be configured to be devoid of data, or to include any data.

According to a 2nd aspect of the invention, there is provided an apparatus configured to prevent rowhammer attack in a computing device. The device has memory which includes a plurality of untrusted memory portions and a plurality of trusted memory portions, the plurality of untrusted memory portions and trusted memory portions being distributed within the memory. The apparatus comprises: a first module arranged to determine respective trusted memory portions that are arranged physically contiguous to respective at least one untrusted memory portion to provide an identified set of trusted memory portions, the respective untrusted memory portions being accessible to perform the attack on the respective trusted memory portions; a second module arranged to determine trusted memory portions in the identified set of trusted memory portions that are movable to a different memory location in the memory; a third

module arranged to move the movable trusted memory portions to an available memory location in the memory to form a block of trusted memory portions; a fourth module arranged to generate at least one memory guard portion; and a fifth module arranged to configure the memory guard portion to be arranged at
5 an end of the block of trusted memory portions that is physically contiguous to an untrusted memory portion. The memory guard portion prevents electrical-interaction between the block of trusted memory portions and the untrusted memory portion to prevent the attack.

10 According to a 3rd aspect of the invention, there is provided a computer implemented method for preventing rowhammer attack in a computing device. The device has memory which includes at least one untrusted memory portion, at least one trusted memory portion and a first data portion, the first data portion accessible via the untrusted memory portion to perform the attack on the trusted
15 memory portion. The method comprises: (i) duplicating contents of the first data portion; (ii) copying the duplicated contents to a second data portion in the memory, the second data portion configured to be isolated from the first data portion; and (iii) configuring the first and second data portions to be accessible only by the trusted memory portion, and the untrusted memory portion
20 respectively.

Preferably, the first data portion may be configured to only enable a read access by the UM portion, and to enable a read access and/or a write access by the TM portion.

25 Preferably, the method may further comprise: (iv) generating at least one memory guard portion; and (v) configuring the memory guard portion to be arranged intermediate the first and second data portions, if the first and second data portions are arranged physically contiguous with each other, wherein the
30 memory guard portion prevents electrical-interaction between the first and second data portions.

Preferably, the method may be configured to be incorporated in a memory allocator and a bootloader of an operating system of the device.

35

Preferably, the trusted memory portion may include being secured against access by unauthorized software processes executed on the device.

5 Preferably, the memory may include dynamic random-access memory or NAND flash memory.

Preferably, the memory guard portion may be configured to be devoid of data, or to include any data.

10 According to a 4th aspect of the invention, there is provided an apparatus configured to prevent rowhammer attack in a computing device. The device has memory which includes at least one untrusted memory portion, at least one trusted memory portion and a first data portion, the first data portion accessible via the untrusted memory portion to perform the attack on the trusted memory
15 portion. The apparatus comprises: a first module arranged to duplicate contents of the first data portion; a second module arranged to copy the duplicated contents to a second data portion in the memory, the second data portion configured to be isolated from the first data portion; and a third module arranged to configure the first and second data portions to be accessible only by the
20 trusted memory portion, and the untrusted memory portion respectively.

It should be apparent that features relating to one aspect of the invention may also be applicable to the other aspects of the invention.

25 These and other aspects of the invention will be apparent from and elucidated with reference to the embodiments described hereinafter.

Brief Description of the Drawings

Embodiments of the invention are disclosed hereinafter with reference to the
30 accompanying drawings, in which:

FIG. 1 shows execution of a rowhammer attack, according to the prior art.

FIG. 2 is a flow diagram of a computer implemented method for preventing rowhammer attack in a computing device, according to a first embodiment.

FIGs. 3a and 3b schematically show application of the method of FIG.2 to the
35 memory of the computing device.

FIGs. 4a and 4b respectively show how trusted memory portions may be moved during boot-time and run-time to form a block of trusted memory portion.

FIG. 5 schematically shows the working mechanism of a memory guard portion.

FIG. 6 is a flow diagram of a computer implemented method for preventing rowhammer attack in a computing device, according to a second embodiment.

FIGs. 7a and 7b schematically show application of the method of FIG.6 to the memory of the computing device.

FIG. 8 is a first example scenario in which the method of FIG.2 is applied to protect a trusted execution environment.

FIG. 9 is a second example scenario in which the method of FIG.2 is applied to protect the kernel of an operating system.

FIG. 10 is a third example scenario in which the method of FIG.2 is applied to protect hypervisors.

FIG. 11 is a fourth example scenario in which the method of FIG.2 is applied to protect software processes.

FIG. 12 shows a schematic summary of the respective methods of FIGs. 2 and 6.

Detailed Description of Preferred Embodiments

FIG. 2 depicts a flow diagram of a computer implemented method 200 (hereafter first method) for preventing rowhammer attack in a computing device (not shown), according to a first embodiment. In one example, the first method 200 is configured to be incorporated in both a memory allocator (MA) and a bootloader (BL) of an operating system of the computing device. The bootloader may also be known as system loader (SL). The computing device has (physical) memory (e.g. dynamic random-access memory or NAND flash memory) which includes a plurality of untrusted memory portions, and a plurality of trusted memory portions, wherein the plurality of untrusted memory portions and trusted memory portions are distributed within the memory. As a brief background, physical memory is typically classified into TM portions and UM portions. Each trusted memory portion is digitally secured against access by unauthorized software processes executed on the computing device. Broadly, the first method 200 comprises: at step 202, determining respective trusted memory portions that are arranged physically contiguous to respective at least one untrusted memory portion to provide an identified set of trusted memory portions, the respective

untrusted memory portions being accessible to perform the attack on the respective trusted memory portions (e.g. modify data stored in the respective trusted memory portions); at step 204, determining trusted memory portions in the identified set of trusted memory portions that are movable to a different memory location in the memory; at step 206, moving the movable trusted memory portions to an available memory location in the memory to form a block of trusted memory portions; at step 208, generating at least one memory guard portion; and at step 210, configuring the memory guard portion to be arranged at an end of the block of trusted memory portions that is physically contiguous to an untrusted memory portion.. Beneficially, the memory guard portion prevents electrical-interaction between the block of trusted memory portions and the untrusted memory portion to prevent the attack. In this manner, rowhammer attacks initiated from the untrusted memory portion causes no malicious effect on the trusted memory portion, since the memory guard portion is empty (and therefore cannot affect the trusted memory portion). It is to be appreciated that the memory guard portion may also be termed as memory boundary guard (MBG).

It is to be appreciated that the first method 200 may also be realized in hardware form as an associated apparatus, and accordingly, the respective steps 202-210 are then implementable as a first module, a second module, a third module, a fourth module, and a fifth module respectively arranged to perform the associated method steps set out above.

FIGs. 3a and 3b schematically show the first method 200 applied to the memory of the computing device. More specifically, comparison of the (physically mapped) memory layout of the memory as conventionally (i.e. FIG. 3a) and as modified by the first method 200 (i.e. FIG. 3b) is depicted. As known in the art, rowhammer attack may be mounted at respective boundaries between physically contiguous adjacent UM and TM portions. In the basic configuration, the first method 200 is configured to generate memory guard portion(s) and to arrange the generated memory guard portion(s) between (physically contiguous) UM and TM portions to prevent the attack. It is to be appreciated that the generated memory guard is arranged to be immediately adjacent to the UM and TM portions. In other words, if the UM and TM portions are not physically

contiguous, then it is not necessary to generate and arrange a memory guard portion between the UM and TM portions.

5 So, it is highlighted that the first method 200 is configured to merge separate TM portions into respective blocks of memory portions, which thereby reduces a number of memory boundaries between TM and UM portions. In turn, this reduces rowhammer locations that may be used to carry out the attack. In relation, FIGs. 4a and 4b respectively show how TM portions may be moved during boot-time and run-time to form respective blocks of trusted memory portions. In one example, the first method 200 (via the BL and MA) is configured to merge all TM portions together, if the TM portions are determined to be movable (i.e. not fixed memory portions). First, at boot time, the SL allocates all TM portions to be arranged together, if possible. Second, at run time then, whenever a new TM portion is generated, the MA is configured to allocate the new TM portion to an unallocated memory portion that has sufficient memory space to accommodate the new TM portion, and which is also immediately adjacent to another existing TM portion.

20 Next, FIG. 5 schematically shows the working mechanism of a memory guard portion. As mentioned, the memory guard portion is configured to be devoid of data, i.e. configured as an empty physical memory portion. Additionally, the memory guard portion is configured to physically belong to the adjacent TM portion. The life cycle of the memory guard portion is also configured to be the same as the life cycle of the adjacent TM portion. To reiterate, the first method 200 (via the MA) only generates a memory guard portion to be arranged between a TM portion and a UM portion that are physically contiguous. The size of the memory guard portion generated is based on memory hardware characteristics of the computing device (in concern). The default size of a memory guard portion to be generated will be based on a row size of the memory hardware of the computing device.

35 Further embodiments of the invention will be described hereinafter. For the sake of brevity, description of like elements, functionalities and operations that are common between the embodiments are not repeated; reference will instead be made to similar parts of the relevant embodiment(s).

FIG. 6 is a flow diagram of a computer implemented method 600 (hereafter second method) for preventing rowhammer attack in a computing device, according to a second embodiment. In one example, the second method 600 is configured to be incorporated in both a memory allocator (MA) and a bootloader (BL) of an operating system of the computing device. The bootloader may also be known as system loader (SL). The memory includes at least one untrusted memory (UM) portion, at least one trusted memory (TM) portion and a first data portion (i.e. see FIG. 7a), in which the first data portion is accessible via the untrusted memory portion to perform the attack on the trusted memory portion.

It is to be appreciated that the first data portion is a read-only shared memory (SM) accessible by both the trusted memory portion and the untrusted memory portion. Particularly, the first data portion is configured to only enable a read access by the UM portion, and to enable a read access and/or a write access by the TM portion. Broadly, the second method 600 comprises: at step 602, duplicating contents of the first data portion; at step 604, copying the duplicated contents to a second data portion in the memory, the second data portion arranged to be physically isolated from the first data portion (i.e. see FIG. 7b); and at step 606, configuring the first and second data portions to be accessible only by the trusted memory portion, and the untrusted memory portion respectively.

The second method 600 may also be realized in hardware form as an associated apparatus, and accordingly, the respective steps 602-606 are then implementable as a first module, a second module, and a third module respectively arranged to perform the associated method steps set out above.

It is to be appreciated that at step 606, the first data portion and the second data portion may collectively be termed as a shared memory guard (SMG), and then the first data portion and the second data portion may also respectively be termed as trusted shared memory guard (TSMG) and untrusted shared memory guard (USMG). The life cycle of the first data portion and second data portion are configured to be the same as the trusted memory portion and untrusted memory portion respectively.

More specifically, the second method 600 (via the MA) generates the SMG as follows. The original SM shared between the UM and TM portions is first converted to the TSMG that is made accessible only by the TM portion. Next, the MA generates a duplicate copy of the TSMG (i.e. designated as the USMG), which is made accessible only by the UM portion. It is to be appreciated that the TSMG and USMG are configured to be physically isolated. Conventionally, the original SM is a rowhammer location, but after processed by the second method 600, the rowhammer location from the original SM is consequently removed (being converted to TSMG and USMG).

Additionally, the second method 600 further comprises: generating at least one memory guard portion configured to be devoid of data; and configuring the memory guard portion to be arranged intermediate the first and second data portions (only if the first and second data portions are arranged physically contiguous with each other). The memory guard portion prevents electrical-interaction between the first and second data portions. It is to be appreciated that this is an optional feature, and not necessary for the second method 600 to function.

FIG. 8 is a first example scenario in which the first method 200 is applied to protect a trusted execution environment (TEE) operating system from a separate operating system, e.g. Linux. The TEE OS will be configured to reside in a first memory portion, and Linux will be configured to reside in a second memory portion that is physically contiguous to the first memory portion. Specifically, the first method 200 (via the BL) loads the TEE OS and maps the TEE OS to the first memory portion. The BL in this case belongs to the TEE OS. Next, the MA of the TEE OS generates and loads a memory boundary portion (devoid of data) to physically isolate the second memory portion (which holds Linux) from the first memory portion (which holds the TEE OS). In this manner, the TEE OS is not affected by rowhammer attacks initiated from Linux. The first memory portion may be viewed as the TM portion, whereas the second memory portion may be viewed as the UM portion in this instance.

FIG. 9 is a second example scenario in which the first method 200 is applied to protect the kernel of an operating system (e.g. Linux) from a user application.

The kernel will be configured to reside in a first memory portion, and the user application will be configured to reside in a second memory portion that is physically contiguous to the first memory portion. Specifically, the BL loads and isolates the kernel. Then, the MA in the kernel loads and physically isolates the user application (from the kernel) by generating a memory boundary portion, which is arranged between the first and second memory portions. It is to be appreciated that the memory boundary portion belongs to the first memory portion. Hence, the first memory portion is not affected by a rowhammer attack launched from the second memory portion. The first memory portion may be viewed as the TM portion, while the second memory portion may be viewed as the UM portion in this instance.

FIG. 10 is a third example scenario in which the first method 200 is applied to protect hypervisors in virtual machine settings. Specifically, respective memory portions that hold a hypervisor, a host OS and a guest OS may be isolated from one another using their respective own memory boundary portions. It is to be appreciated that the MA of the hypervisor or MA of the host OS is usable to generate and manage the memory boundary portion (and the SMG if necessary to be generated).

FIG. 11 is a fourth example scenario in which the first method 200 is applied to protect a plurality of software processes. For example, there are two software processes (e.g. "process 1" and "process 2") depicted in FIG. 11, in which "process 1" will be configured to reside in a first memory portion and "process 2" will be configured to reside in a second memory portion that is physically contiguous to the first memory portion. In this case, "process 1" is to be protected from "process 2", and hence the first memory portion is designated as the TM portion, while the second memory portion is designated as the UM portion. The MA in the OS kernel (hosting "process 1" and "process 2") may be used to generate and manage memory boundary portions belonging to the respective processes. In this way, a trusted process cannot be compromised by rowhammer attack launched from an untrusted process.

In summary, the proposed first method 200 provides a software-based physical memory isolation solution (incorporated in the BL and MA) to effectively protect

TM portions from being modified by rowhammer attack via UM portions. Specifically, by re-arranging the physical memory layout (via the first method 200), memory portions that can be compromised by rowhammer attack are reduced. The physical memory consumption is also optimized. A rowhammer attack comprises performance of the following steps: (i) rowhammer testing, (ii) memory manipulation and (iii) data oriented programming (DOP) attack. The first method 200 is devised to prevent the step of memory manipulation, which then effectively prevents the attack. The second method 600 supports secure sharing of read-only data portions (in memory) between UM portions and TM portions, removing the rowhammer effect on the read-only data portions. FIG. 12 shows a schematic summary of the first and second methods 200, 600.

For the first method 200, when the operating system starts, the BL initially loads and isolates TM and UM portions to physically unrelated portions in the memory. When the operating system is running, the MA further manages the layout of TM portions and UM portions, and generates respective memory boundary guards (MBGs) to isolate different TM portions and UM portions (if they are physically contiguous). When read-only memory is shared between the UM portions and TM portions, the MA generates a SMG (which comprises the TSMG and USMG) as a means to isolate the read-only shared memory. The MA is arranged to manage and dynamically maintain the respective life cycles of the TM portions, UM portions, MBGs and SMGs, universally.

Advantageously, the first method 200 provides a lightweight software solution, which is applicable to existing systems with minimal memory consumption, substantially zero CPU overhead and energy consumption. The second method 600 provides an isolation mechanism versus existing solutions that mostly focus on blocking the memory bit flip. So using the second method 600, even if the physical memory is flipped, the protected memory region is not affected by the bit flip. Specifically, the second method 600 supports read-only shared memory. Most existing solutions do not protect shared memory, causing security problems, or prevent use of shared memory between different entities, resulting in compatibility issues. The second method 600 enables normal memory sharing and blocks rowhammer attack by generating an SMG for usage.

Apart from rowhammer attack, the first and second methods 200, 600 are also applicable to prevent general DOP attacks that can cause privilege escalations. Also, both the first and second methods 200, 600 may be made into corresponding software operating system product.

5

While the invention has been illustrated and described in detail in the drawings and foregoing description, such illustration and description are to be considered illustrative or exemplary, and not restrictive; the invention is not limited to the disclosed embodiments. Other variations to the disclosed embodiments can be understood and effected by those skilled in the art in practising the claimed invention. For example, the memory guard portion need not always be configured to be devoid of data; instead, the memory guard portion may in certain instances also be arranged to include any data.

10

15

Claims

1. A computer implemented method for preventing rowhammer attack in a computing device, the device having memory which includes untrusted memory portions, trusted memory portions and a first data portion, the untrusted memory portions and trusted memory portions being distributed within the memory, the first data portion accessible by a read-only function via the untrusted memory portion and accessible by a read-and-write function via the trusted memory portion, the method comprises:

(i) determining trusted memory portions that are arranged physically contiguous to at least one untrusted memory portion to provide an identified set of trusted memory portions, the untrusted memory portions being accessible to perform the attack on the physically contiguous trusted memory portions;

(ii) determining trusted memory portions in the identified set of trusted memory portions that are movable to a different memory location in the memory;

(iii) moving the movable trusted memory portions to an available memory location in the memory to form a block of trusted memory portions;

(iv) generating at least one memory guard portion;

(v) configuring the memory guard portion to be arranged at an end of the block of trusted memory portions that is physically contiguous to an untrusted memory portion, wherein the memory guard portion prevents electrical-interaction between the block of trusted memory portions and the untrusted memory portion to prevent the attack;

(vi) duplicating contents of the first data portion;

(vii) copying the duplicated contents to a second data portion in the memory, the second data portion configured to be isolated from the first data portion; and

(viii) configuring the first and second data portions to be accessible only by the trusted memory portion, and the untrusted memory portion respectively.

2. The method of claim 1, wherein the method is configured to be incorporated in a memory allocator and a bootloader of an operating system of the device.

3. The method of any preceding claims, wherein each trusted memory portion includes being secured against access by unauthorized software processes executed on the device.

4. The method of any preceding claims, wherein the memory includes dynamic random-access memory or NAND flash memory.

5. The method of any preceding claims, wherein the memory guard portion is configured to be devoid of data, or to include any data.

6. An apparatus configured to prevent rowhammer attack in a computing device, the device having memory which includes untrusted memory portions, trusted memory portions, and a first data portion, the untrusted memory portions and trusted memory portions being distributed within the memory, the first data portion accessible by a read-only function via the untrusted memory portion and accessible by a read-and-write function via the trusted memory portion, the apparatus comprising:

a first module arranged to determine trusted memory portions that are arranged physically contiguous to at least one untrusted memory portion to provide an identified set of trusted memory portions, the untrusted memory portions being accessible to perform the attack on the physically contiguous trusted memory portions;

a second module arranged to determine trusted memory portions in the identified set of trusted memory portions that are movable to a different memory location in the memory;

a third module arranged to move the movable trusted memory portions to an available memory location in the memory to form a block of trusted memory portions;

a fourth module arranged to generate at least one memory guard portion;

a fifth module arranged to configure the memory guard portion to be arranged at an end of the block of trusted memory portions that is physically contiguous to an untrusted memory portion, wherein the memory guard portion prevents electrical-interaction between the block of trusted memory portions and the untrusted memory portion to prevent the attack

a sixth module arranged to duplicate contents of the first data portion;

a seventh module arranged to copy the duplicated contents to a second data portion in the memory, the second data portion configured to be isolated from the first data portion; and

an eighth module arranged to configure the first and second data portions to be accessible only by the trusted memory portion, and the untrusted memory portion respectively.