(12) UK Patent (19) GB (11) 2576755 (13) B

(45) Date of B Publication 06.01.2021

(54) Title of the Invention: System and method for providing protected data storage in a data memory

(51) INT CL: ***G06F 21/78*** (2013.01) ***G06F 12/14*** (2006.01)

(21) Application No: **1814149.9**

(22) Date of Filing: **31.08.2018**

(43) Date of A Publication **04.03.2020**

(56) Documents Cited:

GB 2136175 A
WO 2007/144388 A1
US 8812875 B1
US 20020129245 A1
EP 2674891 A1
WO 1997/026736 A1
US 20040032950 A1

(58) Field of Search:

As for published application 2576755 A viz:
INT CL **G06F**
Other: **EPODOC, WPI, INSPEC, Patent Fulltext, XPESP, XPIEE, XPIPCOM, XPI3E, XPMISC, XPLNCS, XPRD, XPSPRNG, TDB**
updated as appropriate

Additional Fields
INT CL **H04L**
Other: **None**

(72) Inventor(s):
Tuomas Mikael Kärkkäinen

(73) Proprietor(s):
Gurulogic Microsystems Oy
Linnankatu 34, Turku FI-20100, Finland

(74) Agent and/or Address for Service:
Basck Ltd
WeWork, 50-60 Station Road, Cambridge, Cambridgeshire, CB1 2JH, United Kingdom

GB 2576755 B

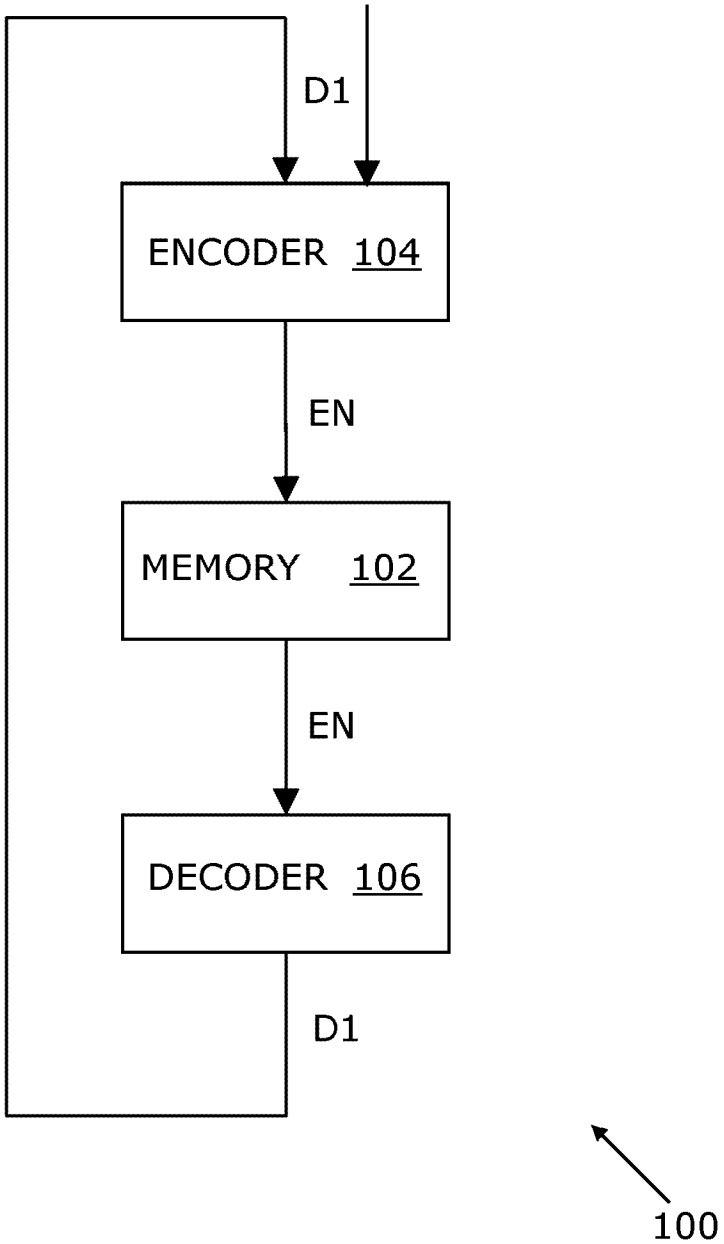

FIG. 1A

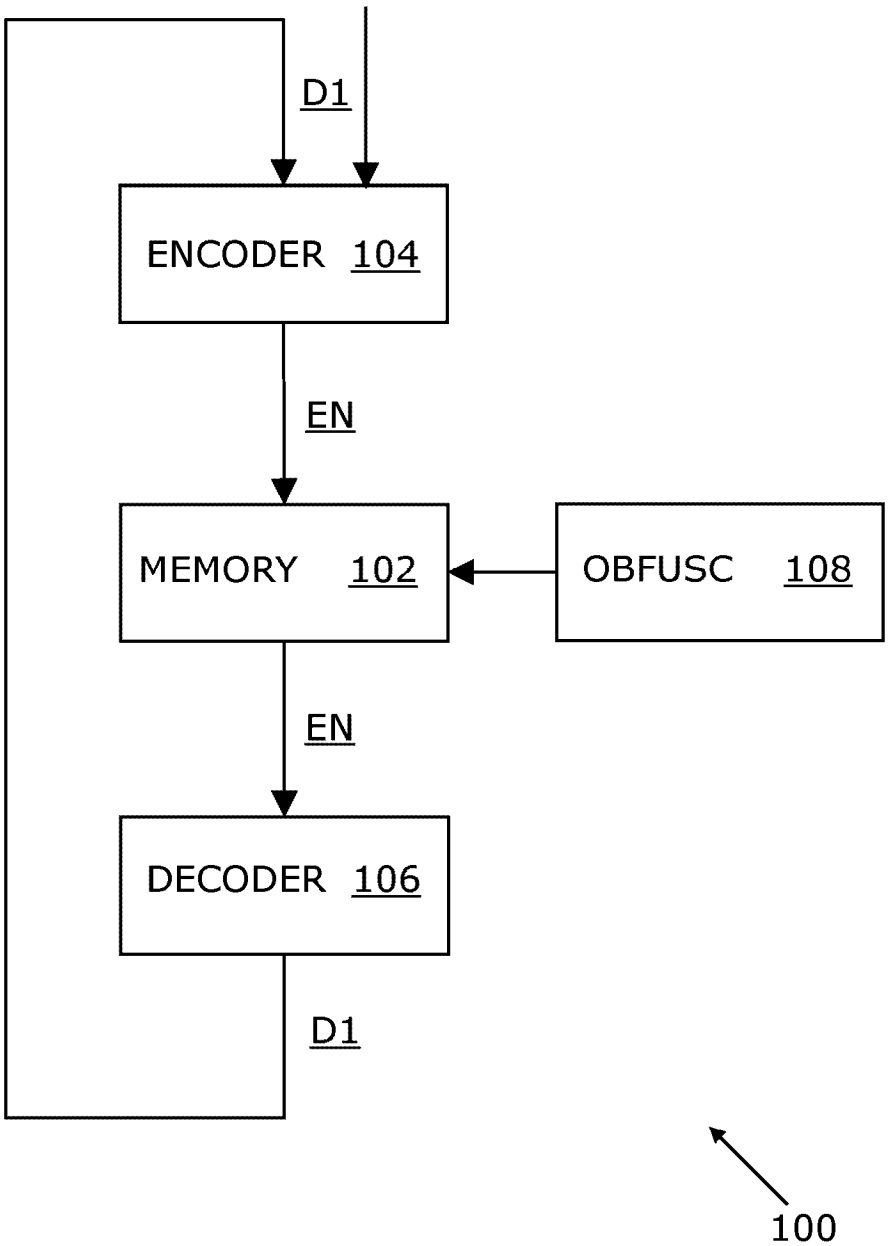

FIG. 1B

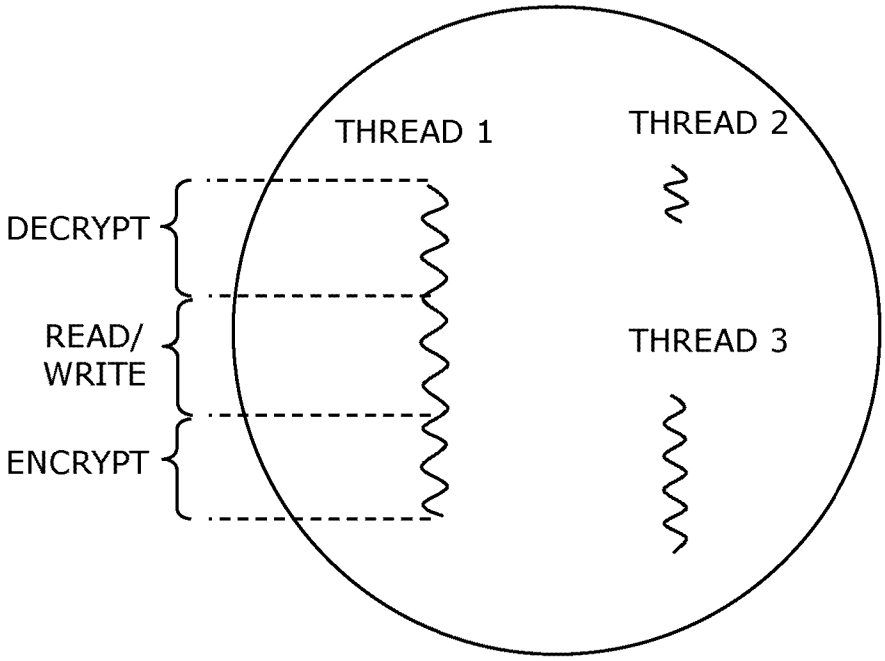

FIG. 2

SYSTEM AND METHOD FOR PROVIDING PROTECTED DATA STORAGE IN DATA MEMORY

TECHNICAL FIELD

The present disclosure relates to systems for providing protected data storage in data memories of computing devices. Moreover, the present disclosure is concerned with methods for providing protected data storage in data memories of computing devices. Furthermore, the present disclosure concerns computer program products comprising non-transitory computer-readable storage media having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute the aforesaid methods.

BACKGROUND

Contemporary computer-related products (for example, such as processing hardware, operating systems and so forth) beneficially conform to data security standards. As an example, for a given processing hardware, it is desired that the given processing hardware has an isolated or trusted environment, where all sensitive data can be processed securely. Providing such an isolated or trusted environment often increases a cost of manufacturing a given computer-related product.

Moreover, it is contemporarily expected that, in many situations, service providers (providing various services to their consumers) execute their duty of care in respect of data right protection of the consumers, and fulfil relevant legal requirements.

Conventionally, hardware vendors have provided a Trusted Execution Environment (TEE), which guarantees "in theory" protection for execution code and sensitive data inside isolated hardware. Moreover, contemporary operating systems (OS) employ a memory protection technique, which prevents a process from accessing a portion of memory that has not been

allocated to it. Furthermore, some conventional techniques encrypt the data prior to storage in the memory. Such data encryption utilizes a fixed encryption key that is selected by a user or by a software application.

However, these conventional techniques for protecting sensitive data suffer from several disadvantages. Firstly, runtime memory is isolated from protected memory by hardware or an operating system (OS). Secondly, the protected memory is static during non-modified memory usage. Thirdly, one cannot trust protection techniques provided by device manufacturers (for example, such as TEE), as security implementations of such protection techniques are not transparent to their users. Fourthly, the conventional techniques are vulnerable to memory attacks, for example, such as "Meltdown" and "Spectre", which are critical vulnerabilities in modern processing hardware. Fifthly, vulnerabilities existing in the modern processing hardware cannot be fixed easily, namely without changing their overall design. Sixthly, memory operations employed at the OS level (to safeguard against the aforementioned vulnerabilities) reduce computing performance severely. Seventhly, one cannot trust protection techniques provided by the operating systems (OS), as their security implementations are typically based on information systems that are comprehensive and intact only in theory; notably, new vulnerabilities are being found from interfaces of such information systems every now and then.

In light of the foregoing, there arises a contemporary need for an improved system for providing protected data storage in a data memory of a computing device, such that the protected data storage is not vulnerable to memory attacks.

SUMMARY

The present disclosure seeks to provide an improved system for providing protected data storage in a data memory of a computing device.

Moreover, the present disclosure seeks to provide an improved method for providing protected data storage in a data memory of a computing device.

A further aim of the present disclosure is to at least partially overcome at least some of the problems of the prior art, as described in the foregoing.

In a first aspect, embodiments of the present disclosure provide a system that, when in operation, provides protected data storage in a data memory of a computing device, characterized in that the system comprises:

- an encoder executing on a processing hardware of the computing device, wherein the encoder, when in operation:
  - generates encryption information according to an encryption algorithm,
  - encrypts unencrypted data (D1) using the encryption information to generate encrypted data (E2),
  - stores the encrypted data (E2) in an allocated portion of the data memory of the computing device; and
  - stores the encryption information in the allocated portion of the data memory or an allocated portion of an additional data memory of the computing device; and

- a decoder executing on the processing hardware of the computing device, wherein the decoder, when in operation:
  - accesses the encrypted data (E2) from the allocated portion of the data memory and the encryption information from the allocated portion of the data memory or the allocated portion of the additional data memory, and
  - decrypts the encrypted data (E2) using the encryption information to re-generate the unencrypted data (D1);

wherein the encoder, when in operation:
  - generates new encryption information according to the encryption algorithm,
  - re-encrypts the unencrypted data (D1) using the new encryption information to generate new encrypted data (E3),
  - replaces the encrypted data (E2) with the new encrypted data (E3) in the allocated portion of the data memory, and

- replaces the encryption information with the new encryption information in the allocated portion of the data memory or the allocated portion of the additional data memory,

wherein the unencrypted data (D1) is re-encrypted using newer encryption information to generate newer encrypted data (EN+1) each time the unencrypted data (D1) is read from the allocated portion of the data memory or the unencrypted data (D1) is to be written to the allocated portion of the data memory, wherein previous encrypted data (EN) is to be replaced with the newer encrypted data (EN+1) in the allocated portion of the data memory,

further wherein the encoder and the decoder are integrated, such that the decoder and the encoder, when in operation, decrypt the previous encrypted data (EN) into the unencrypted data (D1) and re-encrypt the unencrypted data (D1) into the newer encrypted data (EN+1), respectively, in a single thread of execution, and wherein the encoder and the decoder are implemented by way of a low-level code in an inline configuration, such that the processing hardware of the computing device does not interrupt a cycle of decryption and encryption.

Embodiments of the present disclosure are of advantage in that the system provides more robust protected data storage against different kinds of memory attacks, and is not prone to vulnerabilities of operating systems, target platforms and hardware.

In a second aspect, embodiments of the present disclosure provide a method for providing protected data storage in a data memory of a computing device, the method being implemented by a system comprising an encoder and a decoder, characterized in that the method comprises:

- generating, via the encoder, encryption information according to an encryption algorithm;
- encrypting, via the encoder, unencrypted data (D1) using the encryption information to generate encrypted data (E2);

- storing the encrypted data (E2) in an allocated portion of the data memory of the computing device;
- storing the encryption information in the allocated portion of the data memory or an allocated portion of an additional data memory of the computing device
- accessing, via the decoder, the encrypted data (E2) from the allocated portion of the data memory and the encryption information from the allocated portion of the data memory or the allocated portion of the additional data memory;
- decrypting the encrypted data (E2) using the encryption information to re-generate the unencrypted data (D1);
- generating, via the encoder, new encryption information according to the encryption algorithm; and
- re-encrypting, via the encoder, the unencrypted data (D1) using the new encryption information to generate new encrypted data (E3);
- replacing the encrypted data (E2) with the new encrypted data (E3) in the allocated portion of the data memory, and replacing the encryption information with the new encryption information in the allocated portion of the data memory or the allocated portion of the additional data memory;

wherein the steps of generating newer encryption information, re-encrypting the unencrypted data (D1) to generate newer encrypted data (EN+1) and replacing previous encrypted data (EN) with the newer encrypted data (EN+1) are repeated each time the unencrypted data (D1) is read from the allocated portion of the data memory or the unencrypted data (D1) is to be written to the allocated portion of the data memory,

further wherein the encoder and the decoder are integrated, such that the steps of decrypting the previous encrypted data (EN) into the unencrypted data (D1) and re-encrypting the unencrypted data (D1) into the newer encrypted data (EN+1) are performed in a single thread of execution, and wherein the encoder (104) and the decoder (106) are implemented by way of a low-level code in an inline configuration, such that a cycle of decryption and encryption is not interrupted.

In a third aspect, embodiments of the present disclosure provide a computer program product comprising a non-transitory computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute a method pursuant to the aforementioned second aspect.

Additional aspects, advantages, features and objects of the present disclosure would be made apparent from the drawings and the detailed description of the illustrative embodiments construed in conjunction with the appended claims that follow.

It will be appreciated that features of the present disclosure are susceptible to being combined in various combinations without departing from the scope of the present disclosure as defined by the appended claims.

BRIEF DESCRIPTION OF THE DRAWINGS

The summary above, as well as the following detailed description of illustrative embodiments, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the present disclosure, exemplary constructions of the disclosure are shown in the drawings. However, the present disclosure is not limited to specific methods and apparatus disclosed herein. Moreover, those in the art will understand that the drawings are not to scale. Wherever possible, like elements have been indicated by identical numbers.

Embodiments of the present disclosure will now be described, by way of example only, with reference to the following diagrams wherein:

FIGs. 1A and 1B are schematic illustrations of a system for providing protected data storage in a data memory of a computing device, in accordance with different embodiments of the present disclosure; and

FIG. 2 is a schematic illustration of how a cycle of decryption and encryption is performed in a single thread of execution, pursuant to embodiments of the present disclosure.

In the accompanying diagrams, an underlined number is employed to represent an item over which the underlined number is positioned or an item to which the underlined number is adjacent. When a number is non-underlined and accompanied by an associated arrow, the non-underlined number is used to identify a general item at which the arrow is pointing.

DETAILED DESCRIPTION OF EMBODIMENTS

In the following detailed description, illustrative embodiments of the present disclosure and ways in which they can be implemented are elucidated. Although some modes of carrying out the present disclosure are described, those skilled in the art would recognize that other embodiments for carrying out or practising the present disclosure are also possible.

In a first aspect, embodiments of the present disclosure provide a system that, when in operation, provides protected data storage in a data memory of a computing device, characterized in that the system comprises:

- an encoder executing on a processing hardware of the computing device, wherein the encoder, when in operation:
    - generates encryption information according to an encryption algorithm,
    - encrypts unencrypted data (D1) using the encryption information to generate encrypted data (E2),
    - stores the encrypted data (E2) in an allocated portion of the data memory of the computing device; and
    - stores the encryption information in the allocated portion of the data memory or an allocated portion of an additional data memory of the computing device; and

- a decoder executing on the processing hardware of the computing device, wherein the decoder, when in operation:
  - accesses the encrypted data (E2) from the allocated portion of the data memory and the encryption information from the allocated portion of the data memory or the allocated portion of the additional data memory, and
  - decrypts the encrypted data (E2) using the encryption information to re-generate the unencrypted data (D1);

wherein the encoder, when in operation:

- generates new encryption information according to the encryption algorithm,
- re-encrypts the unencrypted data (D1) using the new encryption information to generate new encrypted data (E3),
- replaces the encrypted data (E2) with the new encrypted data (E3) in the allocated portion of the data memory, and
- replaces the encryption information with the new encryption information in the allocated portion of the data memory or the allocated portion of the additional data memory,

wherein the unencrypted data (D1) is re-encrypted using newer encryption information to generate newer encrypted data (EN+1) each time the unencrypted data (D1) is read from the allocated portion of the data memory or the unencrypted data (D1) is to be written to the allocated portion of the data memory, wherein previous encrypted data (EN) is to be replaced with the newer encrypted data (EN+1) in the allocated portion of the data memory,

further wherein the encoder and the decoder are integrated, such that the decoder and the encoder, when in operation, decrypt the previous encrypted data (EN) into the unencrypted data (D1) and re-encrypt the unencrypted data (D1) into the newer encrypted data (EN+1), respectively, in a single thread of execution, and wherein the encoder and the decoder are implemented by way of a low-level code in an inline configuration, such that

the processing hardware of the computing device does not interrupt a cycle of decryption and encryption.

Throughout the present disclosure, the term "thread of execution" generally refers to a smallest sequence of computer-readable instructions that can be executed independently by a scheduler. A thread of execution is a component of a process; herein, the term "process" generally refers to an instance of a computer program that is being executed, namely during a runtime execution of the computer program. While a computer program is merely a passive collection of computer-readable instructions, a process is an actual execution of those instructions. A process may comprise a single thread of execution or multiple threads of execution that execute computer-readable instructions concurrently.

It will be appreciated that prior to encrypting the unencrypted data (D1) for the first time or encrypting the new data (D1), the data is copied (namely, read) to the data memory in an unencrypted form. Immediately after copying the unencrypted data (D1) or the new data (D1) to the data memory, encryption is performed using dynamically-generated encryption information to generate the encrypted data (E2, E3, EN or EN+1) for storage in the allocated portion of the data memory. The encrypted data (E2, E3, EN or EN+1) is then stored in the allocated portion of the data memory pursuant to embodiments of the present disclosure.

Each time the unencrypted data (D1) is required to be read from the allocated portion of the data memory or the new data (D1) is required to be written to the allocated portion of the data memory, the decoder, when in operation, decrypts the previous encrypted data (EN) into the unencrypted data (D1). After the aforesaid read or write operation, the encoder, when in operation, re-encrypts the unencrypted data (D1) into the newer encrypted data (EN+1). As this cycle of decryption and encryption is performed in the single thread of execution pursuant to embodiments of the present disclosure, the encoder and the decoder operate without any interruption (namely, from a beginning to an end) in the single thread of execution. As a result, the data is never stored in the unencrypted form in the data memory. For illustration

purposes only, an example cycle of decryption and encryption has been elucidated in conjunction with FIG. 2.

The aforementioned system provides a solution that at least partially overcomes at least some of the problems of the prior art, and that is independent of Operating Systems (OS's), target platforms and hardware. Moreover, the system pursuant to embodiments of the present disclosure does not require passwords (or similar) for protection.

Moreover, the aforementioned system, when in operation, functions independently in the single thread of execution, wherein a program utilizing the system is executed in a process; the single thread of execution being a component of said process. In operation, the system does not need to utilize functionalities outside the single thread of execution. As a result, user's sensitive data is protected against interception by malicious third parties.

Pursuant to embodiments of the present disclosure, the aforementioned system is suitable for providing protected storage of sensitive data during an execution of a program (for example, a runtime execution of a software application). The encoder of the system, when in operation, re-encrypts the sensitive data using newer encryption information each time the sensitive data is read from or is to be written to the allocated portion of the data memory. The encoder of the system, when in operation, generates the newer encryption information dynamically. Such a dynamic re-encryption prevents unauthorized access to the sensitive data in an efficient manner. It will be appreciated that as the encoder and the decoder operate in the single thread of execution, accessing the sensitive data or tracking any changes in the sensitive data occurring inside the processing hardware is not possible even for a hardware vendor itself. Thus, the aforementioned system is capable of providing various services and software applications executing on the computing device with an extended protection against malwares, cyber spying, and the like. In this regard, the system is capable of protecting user's sensitive data during the runtime execution even between different hardware and software interfaces.

Throughout the present disclosure, the term "sensitive data" refers to data that is required to be protected from unauthorized access to safeguard the privacy or security of an individual or an organization. Protection of sensitive data may be required for legal or ethical reasons, for issues pertaining to personal privacy, or for proprietary considerations. As an example, the aforementioned system is beneficial to use when creating and handling passwords and Personal Identification Numbers (PIN's), likewise personal data.

Throughout the present disclosure, the term "data memory" generally refers to a memory that is used for temporarily storing variables and intermediate results used during a runtime execution of one or more programs. The term "data memory" encompasses both volatile and non-volatile data memories of the computing device. Some examples of the data memory are a Random-Access Memory (RAM) and a Central Processing Unit (CPU) register.

According to an embodiment, given encrypted data (E2, E3, EN or EN+1) is exported from a volatile memory (for example, such as a RAM) to a non-volatile memory (for example, such as a CPU register, a file system, a database or the like), and is imported back to the volatile memory for runtime execution, as and when required.

Pursuant to embodiments of the present disclosure, the aforementioned system is suitable for protecting sensitive variables stored in, for example, a RAM or a CPU register of the computing device during runtime execution of various services or software applications. Such protected sensitive variables are to be used in a manner that is similar to how unprotected variables are used in conventional techniques. As a result, it is not necessary to make changes to a logical development syntax and paradigm of a given program (for example, a software application).

Throughout the present disclosure, the term "variable" generally refers to a storage location in a given data memory that is identified by a memory address, wherein the storage location is referred to by a symbolic name, and contains some known or unknown quantity of information referred to as a

"data value". Throughout the present disclosure, the term "protected variable" refers to a variable whose data value is protected by the aforesaid dynamic re-encryption (namely, re-encryption using dynamically-generated encryption information) prior to storage in the data memory, pursuant to embodiments of the present disclosure. Pursuant to embodiments of the present disclosure, the protected variable contains the data value in an encrypted form (hereinafter, referred to as the "encrypted data value", for the sake of clarity only). It will be appreciated that storing the data value in the encrypted form (namely, the encrypted data value) provides protection against different kinds of memory attacks, which may try to read or modify the stored data value.

It will be appreciated that a given protected variable is always utilized (namely, for read or write operations during the runtime execution) in the unencrypted form (namely, in a form of plaintext). For this purpose, the aforementioned decoder, when in operation, accesses an encrypted data value of the given protected variable from the data memory, and decrypts the encrypted data value using its corresponding encryption information to generate a decrypted, namely an unencrypted data value. This unencrypted data value is then utilized during the runtime execution.

However, after this unencrypted data value is utilized, the aforementioned encoder, when in operation, dynamically generates newer encryption information and re-encrypts the unencrypted data value of the given protected variable using the newer encryption information to generate a newer encrypted data value (namely, for replacing the encrypted data value stored previously in the data memory). Beneficially, the aforesaid decryption and re-encryption are performed in the single thread of execution. Next time when the given protected variable is required to be utilized for read or write operations, the aforementioned decoder, when in operation, accesses this newer encrypted data value from the data memory and decrypts the newer encrypted data value using the newer encryption information to re-generate the unencrypted data value.

This cycle of decryption and encryption is performed for each read or write operation in the single thread of execution, until the given protected variable is no longer required. The cycle of decryption and encryption can be represented as follows:

Step 1: Encrypt an unencrypted data value (using initial encryption information) into an encrypted data value for the first time and store the encrypted data value in the data memory.

Step 2: Access the encrypted data value from the data memory and decrypt the encrypted data value (using the encryption information) to re-generate the unencrypted data value.

Step 3: Utilize the unencrypted data value, as required. If a read operation is performed, the unencrypted data value remains unchanged. If a write operation is performed, the unencrypted data value changes.

Step 4: Encrypt the unencrypted data value (whether changed or unchanged) using dynamically-generated encryption information into a new encrypted data value and replace the previous encrypted data value with the new encrypted data value in the data memory.

The cycle of the steps 2, 3 and 4 is repeated each time the given protected variable is required to be utilized, and is performed in the single thread of execution. The single thread of execution is not allowed to be suspended until the step 4 is performed (namely, until the unencrypted data value is encrypted into the new encrypted data value and the previous encrypted data value is replaced with the new encrypted data value). The aforesaid cycle of the steps 2, 3 and 4 has been illustrated in conjunction with FIG. 2.

According to an embodiment, the aforementioned encoder and the aforementioned decoder are susceptible to being implemented by employing custom-designed digital hardware, for example via use of one or more Application-Specific Integrated Circuits (ASIC's), custom-designed integrated circuits and similar. In such a case, the processing hardware of the computing

device (on which the encoder and the decoder are executed) includes the custom-designed digital hardware.

Optionally, the system is implemented by employing custom-designed digital hardware that is arranged to operate with hardware associated with controlling the data memory of the computing device, such as to provide a hybrid form of data memory hardware.

According to another embodiment, the aforementioned encoder and the aforementioned decoder are implemented, at least in part, by way of encoding instructions and decoding instructions, respectively, in a given program that, when executed by the processing hardware of the computing hardware, performs the aforementioned encryption and decryption operations.

In such a case, the aforementioned system, in operation, eliminates direct dependency on RAM security solutions, and is not prone to vulnerabilities of Operating Systems (OS's), target platforms and hardware (for example, such as Meltdown and Spectre vulnerabilities, which are critical vulnerabilities in modern processing hardware).

It will be appreciated that the aforementioned system is susceptible to be implemented in low cost consumer devices (namely, low cost computing devices) without compromising an overall data security. In other words, it is possible to implement the aforementioned system independent of hardware architecture of the computing device.

Optionally, the processing hardware of the computing device (on which the encoder and the decoder are executed) includes at least one Reduced Instruction Set Computing (RISC) processor that is configured to execute the encoding and decoding instructions as elucidated earlier. Such a RISC processor is capable of performing relatively simpler concatenated operations at a very high speed, thereby providing a shorter temporal window of opportunity for hostile attacks to occur.

Examples of the computing device include, but are not limited to, a smartphone, a Mobile Internet Device (MID), a tablet computer, an Ultra-Mobile Personal Computer (UMPC), a phablet computer, a Personal Digital Assistant (PDA), a web pad, a Personal Computer (PC), a handheld PC, a laptop computer, a desktop computer, a consumer electronics apparatus, a wireless communication apparatus, a scientific measuring apparatus, a military communications equipment, and a video-conferencing equipment.

Furthermore, according to an embodiment, the unencrypted data (D1) comprises unencrypted data values of at least one protected variable that are to be utilized during a runtime execution of a program. The at least one protected variable could, for example, comprise one or more sensitive variables used during the runtime execution of the program. In some implementations, the at least one protected variable includes a single protected variable. In other implementations, the at least one protected variable includes a plurality of protected variables.

Optionally, the allocated portion of the data memory comprises at least one portion of the data memory that is allocated to the at least one protected variable.

During the runtime execution of the program, read and/or write operations may be performed several times on the at least one protected variable (namely, according to the aforementioned cycle of decryption and encryption). Each time the data (D1) is read or is to be written, newer encryption information is generated dynamically and the data (D1) is re-encrypted using the newer encryption information in the single thread of execution. It will be appreciated that the data (D1) is re-encrypted until last encrypted data (EN+1) is generated using last encryption information before the runtime execution of the program is over.

It is essential from a security perspective that the cycle of decryption and encryption are performed without any interruption. Beneficially, the encoder and the decoder are implemented in an inline configuration (namely, an assembler), such that the processing hardware of the computing device (for

example, a central processing unit of the computing device) does not pause (namely, interrupt) an on-going process of the re-encryption of the data (D1). The inline configuration prevents interception by malicious third parties.

In this regard, the steps of generating the newer encryption information and re-encrypting the data (D1) using the newer encryption information are performed at one go without any hardware interruption, so that there is no time window for interception by malicious third parties. This potentially prevents unauthorized parties from performing a timing attack during the runtime execution of the program.

Optionally, in this regard, the program is implemented by way of a low-level code in the inline configuration, such that the processing hardware of the computing device does not pause (namely, interrupt) an on-going process of the re-encryption of the data (D1) until the code is executed completely.

Additionally, optionally, pointers are used for direct memory access to encrypted data values of the at least one protected variable. This ensures that the encrypted data values are neither transferred outside nor replicated during the runtime execution of the program (for example, a software application). It will be appreciated that the use of the pointers is more secure as compared to memory copy.

It will also be appreciated that reading and writing the encrypted data (E2, E3, EN or EN+1) from and to the allocated portion of the data memory pursuant to embodiments of the present disclosure are performed in a manner that is similar to reading and writing unencrypted data in conventional techniques. In other words, reading and writing the encrypted data values of the at least one protected variable pursuant to embodiments of the present disclosure is performed in a manner that is similar to reading and writing unencrypted data values of a variable in the conventional techniques.

Moreover, optionally, the encoder, when in operation, initializes the encryption information prior to encrypting the unencrypted data (D1) for the

first time. Optionally, in this regard, the encoder, when in operation, generates the encryption information initially (namely, only once) from at least one initialization value. Optionally, the at least one initialization value comprises at least one default value and/or at least one random value. The at least one initialization value is stored in its corresponding allocated portion of the data memory or an additional data memory on a temporary basis.

Optionally, the system further comprises an obfuscation module executing on the processing hardware of the computing device, wherein the obfuscation module, when in operation, obfuscates the at least one initialization value stored in the data memory or the additional data memory prior to releasing its corresponding allocated portion of the data memory or the additional data memory. Such obfuscation may, for example, be performed by way of various types of bit swaps. This ensures that the data memory or the additional data memory (for example, RAM, CPU register or the like) does not store any data reference related to the at least one initialization value that was stored during the memory usage.

Pursuant to embodiments of the present disclosure, the at least one initialization value is used only internally by the aforementioned encoder for generating the encryption information, and is restricted, so that it cannot be accessed or used from outside the aforementioned encoder. In other words, the at least one initialization value is used only internally in the encryption algorithm, in order to generate the encryption information for the first time.

Optionally, the obfuscation module, when in operation, obfuscates the last encrypted data (EN+1) stored in the allocated portion of the data memory prior to releasing the allocated portion of the data memory.

Optionally, in such a case, the obfuscation module, when in operation, obfuscates the last encryption information stored in the allocated portion of the data memory prior to releasing the allocated portion of the data memory.

According to another embodiment, the additional data memory is different from the data memory used to store given encrypted data (E2, E3, EN or EN+1).

Optionally, in such a case, the obfuscation module, when in operation obfuscates the last encryption information stored in the allocated portion of the additional data memory prior to releasing the allocated portion of the additional data memory.

Moreover, optionally, the encoder, when in operation, generates the encryption information randomly. Optionally, in this regard, the encryption information is generated using an automated function. More optionally, new encryption information is generated independent of old encryption information (namely, previously-used encryption information). This eliminates any possibility of creation of an intentional or un-intentional vulnerability by a software developer.

As mentioned earlier, the encryption information is generated according to the encryption algorithm that is to be used for encrypting the unencrypted data (D1). It will be appreciated that how the encryption information is utilized depends on the encryption algorithm.

Optionally, the encryption algorithm employs symmetric encryption. Optionally, in this regard, the encryption algorithm is a block cipher algorithm (see https://en.wikipedia.org/wiki/Block_cipher), for example, such as Advanced Encryption Standard (AES). Alternatively, optionally, the encryption algorithm is a stream cipher algorithm (see https://en.wikipedia.org/wiki/Stream_cipher), for example, such as ChaCha20 algorithm.

It is well known that the ChaCha20 algorithm is a symmetric encryption algorithm with a randomly-generated encryption key and a random integer 'Nonce'. In an example implementation, when the ChaCha20 algorithm is used, first encryption information can be generated from the randomly-generated encryption key and the random integer 'Nonce', such that the first

encryption information has a high entropy. It will be appreciated that in such an case, the randomly-generated encryption key and the random integer 'Nonce' collectively constitute the aforementioned initialization value (namely, the at least one initialization value from which the first encryption information is generated). In the example implementation, subsequent encryption information can be generated using an automated function.

It will be appreciated that the encryption algorithm can alternatively be a suitable asymmetric encryption technique (for example, such as RSA).

According to an embodiment, the encryption information comprises at least one key to be used to encrypt the unencrypted data values of the at least one protected variable to generate the encrypted data values and/or to decrypt the encrypted data values to re-generate the unencrypted data values of the at least one protected variable. In some implementations, the at least one key comprises a single large key. In other implementations, the at least one key comprises a plurality of keys.

According to another embodiment, the encryption information comprises an index of the at least one key to be used to encrypt the unencrypted data values to generate the encrypted data values and/or to decrypt the encrypted data values to re-generate the unencrypted data values. Optionally, in such a case, the at least one key is to be generated by or accessed from a key store using the index. Optionally, the key store is provided to the computing device by a trusted service provider.

Optionally, indices are ordinal numbers of keys in an order of their occurrence within the key store. Optionally, in this regard, the indices are pre-stored in the key store together with their associated keys. Alternatively, optionally, the indices are generated in the key store, and then associated with their respective keys. As an example, the indices may be generated in a consecutive manner corresponding to an order in which the keys are stored in the key store.

Optionally, the key store is protected, and the keys are made accessible for use, internally within the protected key store, to at least one key-store integrated software application, which accesses the keys for use via their indices only. In other words, the keys are not accessible by software applications or ecosystem processes from outside of the key store.

Optionally, the encryption information further comprises a unique identifier of the key store from which the at least one key is to be generated or accessed. This is particularly beneficial when there are a plurality of key stores, and it is important to identify uniquely the key store from which the at least one key is to be generated or accessed.

Optionally, the unique identifier of the key store is a serial number assigned to the key store.

Optionally, the key store is implemented by way of a key container or a key generator that is capable of storing keys and/or generating keys based upon their indices in a reproducible manner. By `reproducible_, it is meant that a same key is generated from a given index in a repeatable manner (namely, in a manner that the key generator always produces the same key with the same index). As an example, the key store can be implemented as described in a UK patent document GB2538052. As another example, the key store can be implemented as described in a UK patent document GB GB2556638.

Optionally, the encoder, when in operation, selects the index randomly, and uses the key store to generate the at least one key based upon the selected index.

For illustration purposes only, there will now be described an example implementation of the aforementioned system for providing protected data storage of data values of a given protected variable used in an example program (for example, a software application). There will now be considered three stages of a runtime execution of the example program.

Phase A:

Before use, the data memory must be internally initialized. Typically, the program is written in a manner that the internal initialization is taken care of automatically. For example, when a given developer writes the program, the given protected variable is uninitialized.

Step A1: A required portion (namely, a size) of the data memory is allocated for storing encrypted data values of the given protected variable. The required portion of the data memory is optionally equal to a size defined for the given protected variable in the example program. Optionally, the allocated portion of the data memory is set to a predefined default value of the given protected variable.

It will be appreciated that the size of the allocated portion of the data memory needs to be equal to or greater than the size defined for the given protected variable. From a technical point of view, it is advantageous when the size of the allocated portion of the data memory is greater than the size defined for the given protected variable, because, in such a case, the size of the allocated portion does not reveal the size defined for the given protected variable.

Step A2: A required portion (namely, a size) of the data memory is allocated for storing encryption information. Optionally, the encryption information is generated dynamically for a defined encryption algorithm. More optionally, the encryption information is generated randomly.

Optionally, the encryption information is generated initially from at least one initialization value, as described earlier. In such a case, the at least one initialization value is stored in its corresponding allocated portion of the data memory or an additional data memory, and is obfuscated (for example, by performing various types of bit swaps) before the corresponding allocated portion of the data memory or the additional data memory is released.

Phase B:

After the allocated portion of the data memory is initialized, the encrypted data values of the given protected variable can be read from or written to the allocated portion of the data memory.

Step B1: In operation, the aforementioned decoder accesses an encrypted data value of the given protected variable from the allocated portion of the data memory, and decrypts the encrypted data value (namely, in a form of ciphertext) to generate a decrypted, namely an unencrypted data value (namely, in a form of plaintext).

ð If the decryption is performed after the step A2, the encryption algorithm is prepared for encryption and decryption purposes, and the encrypted data value is decrypted into the unencrypted data value using the encryption information generated during initialization.

ð If the decryption is performed after the step B3, the encryption algorithm is already being used for encryption and decryption purposes, and the encrypted data value is decrypted into the unencrypted data value using newer encryption information that is re-generated dynamically during encoding (namely, encrypting).

Step B2: The unencrypted data value is utilized for read or write operations during the runtime execution of the program. Optionally, in this regard, the unencrypted data value is returned to a calling function of the program during the runtime execution.

Step B3: In operation, the aforementioned encoder re-generates new encryption information, encrypts the unencrypted data value (namely, in a form of plaintext) using the new encryption information to generate a newly-encrypted data value (namely, in a form of ciphertext). The encrypted data value stored previously is then replaced with the newly-encrypted data value in the allocated portion of the data memory.

It will be appreciated that the encryption information is re-generated dynamically after each encoding iteration; therefore, there is no need to perform time-consuming initialization operations (as performed in the aforementioned step A2) again.

As mentioned earlier, it is essential from the security perspective that a cycle of the aforesaid steps B1, B2 and B3 is performed in a single thread of

execution (namely, at one go without any hardware interruption), so that there is no time window for interception by a malicious third party.

Phase C:

When the given protected variable is not required to be used any more in future, the allocated portion of the data memory is internally finalized. Typically, the program is written in a manner that the internal finalization is taken care of automatically. For example, when a given developer writes the program, the given protected variable is uninitialized and the allocated portion of the data memory is freed.

Optionally, last encryption information stored in its allocated portion of the data memory is obfuscated, for example by performing various types of bit swaps, before the allocated portion of the data memory is released. This ensures that the data memory (for example, RAM, CPU register or the like) does not store any data reference related to the last encryption information that was stored during the memory usage.

Optionally, a last encrypted data value stored in the allocated portion of the data memory is obfuscated before the allocated portion of the data memory is released.

Upon successful completion of the phase C, the allocated portion of the data memory is internally finalized, and is available for use to other programs.

Furthermore, for illustration purposes only, there will now be considered some example cases indicating how re-generated encrypted information and re-encrypted data are stored each time the data (D1) is read or is to be written to the allocated portion of the data memory during a runtime execution of a program. In other words, the encryption information of the data to be encrypted always changes when the data is being handled, namely read or written, during the runtime execution of the program.

In these examples, an example protected variable is allocated 32 bytes in the data memory (hereinafter referred to as a first allocated portion for the sake

of convenience only) and encryption information is also allocated 32 bytes in the data memory (hereinafter referred to as a second allocated portion for the sake of convenience only).

An initial unencrypted data value of the example protected variable (namely, as copied to the data memory for a first time) can be represented as follows:

84, 0, 101, 0, 115, 0, 116, 0, 105, 0, 110, 0, 103, 0, 0, 0, 133, 136, 38, 156, 203, 59, 74, 241, 229, 48, 145, 79, 145, 121, 110, 77

First encryption information is generated according to an encryption algorithm employed, and is then stored in the second allocated portion of the data memory. As an example, according to the ChaCha20 encryption algorithm, the first encryption information can be generated from a randomly-generated encryption key and a random integer 'Nonce', such that the first encryption information has a high entropy.

The first encryption information can be represented as follows:

62, 60, 2, 2, 2, 150, 64, 74, 209, 139, 87, 136, 98, 230, 205, 9, 207, 121, 195, 172, 90, 116, 219, 136, 139, 125, 16, 147, 210, 198, 142, 12

The initial unencrypted data value is then encrypted using the first encryption information to generate a first encrypted data value of the example protected variable, wherein the first encrypted data value is stored in the first allocated portion of the data memory. The first encrypted data value can be represented as follows:

29, 48, 233, 223, 69, 12, 41, 180, 202, 230, 171, 145, 235, 25, 196, 236, 105, 253, 159, 71, 82, 79, 131, 222, 213, 61, 62, 241, 66, 59, 71, 191

Case A: Decrypting the first encrypted data value

The case A concerns the aforementioned step B1, wherein the first encrypted data value of the example protected variable is accessed from the first allocated portion of the data memory, and is decrypted using the

aforementioned first encryption information (stored previously in the second allocated portion of the data memory) to re-generate the initial unencrypted data value of the example protected variable as follows:

84, 0, 101, 0, 115, 0, 116, 0, 105, 0, 110, 0, 103, 0, 0, 0, 133, 136, 38, 156, 203, 59, 74, 241, 229, 48, 145, 79, 145, 121, 110, 77

Case B: Modifying the unencrypted data value

The case B concerns the aforementioned step B2, wherein a first byte of the initial unencrypted data value of the example protected variable is changed from '84' to '1'; and the modified unencrypted data value to be written can be represented as follows:

1, 0, 101, 0, 115, 0, 116, 0, 105, 0, 110, 0, 103, 0, 0, 0, 133, 136, 38, 156, 203, 59, 74, 241, 229, 48, 145, 79, 145, 121, 110, 77

Case C: Encrypting the modified unencrypted data value

The case C concerns the aforementioned step B3, wherein second encryption information is generated dynamically. The second encryption information can be represented as follows:

60, 62, 148, 66, 72, 71, 203, 29, 89, 233, 177, 69, 107, 41, 180, 202, 99, 35, 183, 119, 210, 255, 166, 152, 24, 175, 214, 29, 222, 250, 176, 152

The second encryption information replaces the first encryption information stored previously in the second allocated portion of the data memory. The second encryption information is used to encrypt the modified unencrypted data value to generate a second encrypted data value for storage in the first allocated portion of the data memory. The second encrypted data value can be represented as follows:

61, 62, 241, 66, 59, 71, 191, 29, 48, 233, 223, 69, 12, 41, 180, 202, 230, 171, 145, 235, 25, 196, 236, 105, 253, 159, 71, 82, 79, 131, 222, 213

Case D: Reading a decrypted data value for a first time

In the case D, the second encrypted data value of the example protected variable is accessed from the first allocated portion of the data memory, and decrypted using the second encryption information stored in the second allocated portion of the data memory to re-generate the unencrypted data value as follows:

1, 0, 101, 0, 115, 0, 116, 0, 105, 0, 110, 0, 103, 0, 0, 0, 133, 136, 38, 156, 203, 59, 74, 241, 229, 48, 145, 79, 145, 121, 110, 77

Case E: Re-encrypting the same unencrypted data value

In the case E, in order to encrypt the same unencrypted data value, third encryption information is generated dynamically; the third encryption information can be represented as follows:

126, 118, 211, 137, 85, 30, 34, 172, 28, 130, 152, 241, 161, 74, 151, 125, 20, 241, 72, 209, 74, 231, 9, 78, 5, 113, 44, 173, 70, 132, 198, 75

The third encryption information replaces the second encryption information stored previously in the second allocated portion of the data memory. The third encryption information is used to encrypt the same unencrypted data value to generate a third encrypted data value of the example protected variable for storage in the first allocated portion of the data memory. The third encrypted data value can be represented as follows:

127, 118, 182, 137, 38, 30, 86, 172, 117, 130, 246, 241, 198, 74, 151, 125, 145, 121, 110, 77, 129, 220, 67, 191, 224, 65, 189, 226, 215, 253, 168, 6

Case F: Reading the decrypted data value for a second time

In the case F, the third encrypted data value of the example protected variable is accessed from the first allocated portion of the data memory, and decrypted using the third encryption information stored in the second

allocated portion of the data memory to re-generate the unencrypted data value as follows:

1, 0, 101, 0, 115, 0, 116, 0, 105, 0, 110, 0, 103, 0, 0, 0, 133, 136, 38, 156, 203, 59, 74, 241, 229, 48, 145, 79, 145, 121, 110, 77

Case G: Re-encrypting the same unencrypted data value

In the case G, in order to encrypt the same unencrypted data value, fourth encryption information is generated dynamically; the fourth encryption information can be represented as follows:

247, 35, 205, 171, 249, 2, 160, 52, 237, 35, 210, 102, 220, 94, 102, 53, 197, 187, 175, 216, 4, 226, 120, 98, 168, 55, 168, 107, 13, 115, 229, 134

The fourth encryption information replaces the third encryption information stored previously in the second allocated portion of the data memory. The fourth encryption information is used to encrypt the same unencrypted data value to generate a fourth encrypted data value of the example protected variable for storage in the first allocated portion of the data memory. The fourth encrypted data value can be represented as follows:

246, 35, 168, 171, 138, 2, 212, 52, 132, 35, 188, 102, 187, 94, 102, 53, 64, 51, 137, 68, 207, 217, 50, 147, 77, 7, 57, 36, 156, 10, 139, 203

It will be appreciated that the aforesaid decryption and encryption operations are performed in a repeating manner, without any interruption, in a single thread of execution during the runtime execution of the program. An example cycle of decryption and encryption has been illustrated in conjunction with FIG. 2.

In the above examples, the encryption information to be stored (namely, replacing any previously-stored encryption information) in the second allocated portion of the data memory is shown as underlined text for the sake of clarity only. Likewise, the encrypted data value to be stored (namely,

replacing any previously-stored encrypted data value) in the first allocated portion of the data memory is shown as bold text for the sake of clarity only.

In a second aspect, embodiments of the present disclosure provide a method of (namely, a method for) providing protected data storage in a data memory of a computing device, the method being implemented by a system comprising an encoder and a decoder, characterized in that the method comprises:

- generating, via the encoder, encryption information according to an encryption algorithm;
- encrypting, via the encoder, unencrypted data (D1) using the encryption information to generate encrypted data (E2) and storing the encrypted data (E2) in an allocated portion of the data memory of the computing device;
- accessing, via the decoder, the encrypted data (E2) from the allocated portion of the data memory and decrypting the encrypted data (E2) using the encryption information to re-generate the unencrypted data (D1);
- generating, via the encoder, new encryption information according to the encryption algorithm; and
- re-encrypting, via the encoder, the unencrypted data (D1) using the new encryption information to generate new encrypted data (E3) and replacing the encrypted data (E2) with the new encrypted data (E3) and the encryption information with the new encryption information in the allocated portion of the data memory,

wherein the steps of generating newer encryption information, re-encrypting the unencrypted data (D1) to generate newer encrypted data (EN+1) and replacing previous encrypted data (EN) with the newer encrypted data (EN+1) are repeated each time the unencrypted data (D1) is read from the allocated portion of the data memory or the unencrypted data (D1) is to be written to the allocated portion of the data memory,

further wherein the encoder and the decoder are integrated, such that the steps of decrypting the previous encrypted data (EN) into the unencrypted

data (D1) and re-encrypting the unencrypted data (D1) into the newer encrypted data (EN+1) are performed in a single thread of execution.

Various embodiments and variants disclosed above apply mutatis mutandis to the method.

Optionally, the method further comprises generating initially (namely, only once), via the encoder, the encryption information from at least one initialization value, as described earlier. Optionally, in such a case, the at least one initialization value is stored in its corresponding allocated portion of the data memory or an additional data memory on a temporary basis.

Optionally, the method further comprises obfuscating the at least one initialization value stored in the data memory or the additional data memory prior to releasing its corresponding allocated portion of the data memory or the additional data memory.

Optionally, the method further comprises obfuscating last encrypted data (EN+1) stored in the allocated portion of the data memory prior to releasing the allocated portion of the data memory.

Moreover, the method further comprises storing, via the encoder, given encryption information together with given encrypted data (E2, E3, EN or EN+1) in the allocated portion of the data memory. Optionally, in such a case, the method further comprises obfuscating last encryption information stored in the allocated portion of the data memory prior to releasing the allocated portion of the data memory.

According to another embodiment, the additional data memory is different from the data memory used to store given encrypted data (E2, E3, EN or EN+1). Optionally, in such a case, the method further comprises obfuscating the last encryption information stored in the allocated portion of the additional data memory prior to releasing the allocated portion of the additional data memory.

Optionally, in the method, the unencrypted data (D1) comprises unencrypted data values of at least one protected variable that are to be utilized during a runtime execution of a program.

Optionally, the encryption information comprises at least one key to be used to encrypt the unencrypted data values to generate encrypted data values and/or to decrypt the encrypted data values to re-generate the unencrypted data values.

Alternatively, optionally, the encryption information comprises an index of the at least one key to be used to encrypt the unencrypted data values to generate the encrypted data values and/or to decrypt the encrypted data values to re-generate the unencrypted data values. Optionally, in such a case, the method further comprises generating by or accessing from a key store the at least one key using the index.

Optionally, in the method, the encryption information is generated randomly.

In a third aspect, embodiments of the present disclosure provide a computer program product comprising a non-transitory computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute a method pursuant to the aforementioned second aspect.

Optionally, the computer-readable instructions are downloadable from a software application store, for example, from an `App store_ to the computerized device.

Next, embodiments of the present disclosure will be described with reference to FIGs. 1A and 1B.

FIGs. 1A and 1B are schematic illustrations of a system 100 for providing protected data storage in a data memory 102 of a computing device, in accordance with different embodiments of the present disclosure. Optionally, the system 100 is implemented as custom-designed digital hardware that is

arranged to operate with hardware associated with controlling the data memory 102, such as to provide a hybrid form of data memory hardware.

With reference to FIGs. 1A and 1B, the system 100 comprises an encoder 104 and a decoder 106. The encoder 104, when in operation, re-encrypts unencrypted data (D1) using newly-generated encryption information to generate new encrypted data (EN) each time the unencrypted data (D1) is read from an allocated portion of the data memory 102 or new data (D1) is to be written to the allocated portion of the data memory 102. The encoder 104, when in operation, stores the new encrypted data (EN) in the allocated portion of the data memory 102. The decoder 106, when in operation, accesses the previous encrypted data (EN) from the allocated portion of the data memory, and decrypts the previous encrypted data (EN) using the encryption information to re-generate the unencrypted data (D1). After the unencrypted data (D1) is utilized (namely, for a read or write operation), the encoder 104, when in operation, re-encrypts the unencrypted data (D1) into newer encrypted data (EN+1), and replaces the previous encrypted data (EN) with the newer encrypted data (EN+1).

The encoder 104 and the decoder 106 are integrated, such that the decoder 106 and the encoder 104, when in operation, decrypt the previous encrypted data (EN) into the unencrypted data (D1) and re-encrypt the unencrypted data (D1) into the newer encrypted data (EN+1), respectively, in a single thread of execution.

With reference to FIG. 1B, the system 100 optionally comprises an obfuscation module 108. The obfuscation module 108, when in operation, obfuscates at least one initialization value (used to generate the encryption information initially) stored in its corresponding allocated portion of the data memory 102, prior to releasing the allocated portion of the data memory 102.

FIGs. 1A and 1B are merely examples, which should not unduly limit the scope of the claims herein. A person skilled in the art will recognize many

variations, alternatives, and modifications of embodiments of the present disclosure.

FIG. 2 is a schematic illustration of how a cycle of decryption and encryption is performed in a single thread of execution, pursuant to embodiments of the present disclosure.

As shown, the cycle of decryption and encryption is performed in a single thread of execution 'THREAD 1' as follows:

Step 1: Access an encrypted data value of a protected variable from data memory, and decrypt the encrypted data value (using encryption information) to generate a decrypted data value, namely an unencrypted data value.

Step 2: Utilize the unencrypted data value for a read or write operation.

Step 3: Encrypt the unencrypted data value (whether changed or unchanged) using dynamically-generated encryption information into a new encrypted data value, and replace the aforesaid encrypted data value with the new encrypted data value in the data memory.

The thread 'THREAD 1' is not allowed to be suspended until the step 3 is performed (namely, until the unencrypted data value is encrypted into the new encrypted data value and the previous encrypted data value is replaced with the new encrypted data value).

With reference to FIG. 2, the thread 'THREAD 1' is a component of a process that has multiple threads, for example, such as threads 'THREAD 2' and 'THREAD 3'.

FIG. 2 is merely an example, which should not unduly limit the scope of the claims herein. A person skilled in the art will recognize many variations, alternatives, and modifications of embodiments of the present disclosure. For example, the process may have only one thread of execution, namely the thread 'THREAD 1'.

Modifications to embodiments of the present disclosure described in the foregoing are possible without departing from the scope of the present disclosure as defined by the accompanying claims. Expressions such as `including_, `comprising_, `incorporating_, `consisting of_, `have_, `is_ used to describe and claim the present invention are intended to be construed in a non-exclusive manner, namely allowing for items, components or elements not explicitly described also to be present. Reference to the singular is also to be construed to relate to the plural; as an example, `at least one of_ indicates `one of_ in an example, and `a plurality of_ in another example; moreover, `one or more_ is to be construed in a likewise manner.

The phrases `in an embodiment_, `according to an embodiment_ and the like generally mean the particular feature, structure, or characteristic following the phrase is included in at least one embodiment of the present disclosure, and may be included in more than one embodiment of the present disclosure. Importantly, such phrases do not necessarily refer to the same embodiment.

If the specification states a component or feature `may_, `can_, `could_, or `might_ be included or have a characteristic, that particular component or feature is not required to be included or have the characteristic.

Claims

1. A system that, when in operation, provides protected data storage in a data memory of a computing device, characterized in that the system comprises:

- an encoder executing on a processing hardware of the computing device, wherein the encoder, when in operation:
  - generates encryption information according to an encryption algorithm,
  - encrypts unencrypted data (D1) using the encryption information to generate encrypted data (E2),
  - stores the encrypted data (E2) in an allocated portion of the data memory of the computing device; and
  - stores the encryption information in the allocated portion of the data memory or an allocated portion of an additional data memory of the computing device; and

- a decoder executing on the processing hardware of the computing device, wherein the decoder, when in operation:
  - accesses the encrypted data (E2) from the allocated portion of the data memory and the encryption information from the allocated portion of the data memory or the allocated portion of the additional data memory, and
  - decrypts the encrypted data (E2) using the encryption information to re-generate the unencrypted data (D1);

wherein the encoder, when in operation:
  - generates new encryption information according to the encryption algorithm,
  - re-encrypts the unencrypted data (D1) using the new encryption information to generate new encrypted data (E3),
  - replaces the encrypted data (E2) with the new encrypted data (E3) in the allocated portion of the data memory, and

- replaces the encryption information with the new encryption information in the allocated portion of the data memory or the allocated portion of the additional data memory,

wherein the unencrypted data (D1) is re-encrypted using newer encryption information to generate newer encrypted data (EN+1) each time the unencrypted data (D1) is read from the allocated portion of the data memory or the unencrypted data (D1) is to be written to the allocated portion of the data memory, wherein previous encrypted data (EN) is to be replaced with the newer encrypted data (EN+1) in the allocated portion of the data memory,

further wherein the encoder and the decoder are integrated, such that the decoder and the encoder, when in operation, decrypt the previous encrypted data (EN) into the unencrypted data (D1) and re-encrypt the unencrypted data (D1) into the newer encrypted data (EN+1), respectively, in a single thread of execution, and wherein the encoder and the decoder are implemented by way of a low-level code in an inline configuration, such that the processing hardware of the computing device does not interrupt a cycle of decryption and encryption.

2. A system of claim 1, characterized in that the encoder, when in operation, generates the encryption information initially from at least one initialization value, and the system further comprises an obfuscation module executing on the processing hardware of the computing device, wherein the obfuscation module, when in operation, obfuscates the at least one initialization value stored in its corresponding allocated portion of the data memory or an additional data memory prior to releasing the corresponding allocated portion of the data memory or the additional data memory.

3. A system of claim 1 or 2, characterized in that the additional data memory is different from the data memory used to store given encrypted data (E2, E3, EN or EN+1).

4. A system of any one of claims 1 to 3, characterized in that the unencrypted data (D1) comprises unencrypted data values of at least one protected variable that are to be utilized during a runtime execution of a program, and the encryption information comprises at least one key to be used to encrypt the unencrypted data values to generate

encrypted data values and/or to decrypt the encrypted data values to re-generate the unencrypted data values.

5. A system of any one of claims 1 to 3, characterized in that the unencrypted data (D1) comprises unencrypted data values of at least one protected variable that are to be utilized during a runtime execution of a program, and the encryption information comprises an index of at least one key to be used to encrypt the unencrypted data values to generate encrypted data values and/or to decrypt the encrypted data values to re-generate the unencrypted data values, wherein the at least one key is to be generated by or accessed from a key store using the index.

6. A system of claim 5, characterized in that the encryption information further comprises a unique identifier of the key store from which the at least one key is to be generated or accessed.

7. A system of any one of claims 1 to 6, characterized in that the encoder, when in operation, generates the encryption information randomly.

8. A method for providing protected data storage in a data memory of a computing device, the method being implemented by a system comprising an encoder and a decoder, characterized in that the method comprises:

- generating, via the encoder, encryption information according to an encryption algorithm;
- encrypting, via the encoder, unencrypted data (D1) using the encryption information to generate encrypted data (E2);
- storing the encrypted data (E2) in an allocated portion of the data memory of the computing device;
- storing the encryption information in the allocated portion of the data memory or an allocated portion of an additional data memory of the computing device
- accessing, via the decoder, the encrypted data (E2) from the allocated portion of the data memory and the encryption information from the allocated portion of the data memory or the allocated portion of the additional data memory;
- decrypting the encrypted data (E2) using the encryption information to re-generate the unencrypted data (D1);

- generating, via the encoder, new encryption information according to the encryption algorithm; and
- re-encrypting, via the encoder, the unencrypted data (D1) using the new encryption information to generate new encrypted data (E3);
- replacing the encrypted data (E2) with the new encrypted data (E3) in the allocated portion of the data memory, and replacing the encryption information with the new encryption information in the allocated portion of the data memory or the allocated portion of the additional data memory;

wherein the steps of generating newer encryption information, re-encrypting the unencrypted data (D1) to generate newer encrypted data (EN+1) and replacing previous encrypted data (EN) with the newer encrypted data (EN+1) are repeated each time the unencrypted data (D1) is read from the allocated portion of the data memory or the unencrypted data (D1) is to be written to the allocated portion of the data memory,

further wherein the encoder and the decoder are integrated, such that the steps of decrypting the previous encrypted data (EN) into the unencrypted data (D1) and re-encrypting the unencrypted data (D1) into the newer encrypted data (EN+1) are performed in a single thread of execution, and wherein the encoder (104) and the decoder (106) are implemented by way of a low-level code in an inline configuration, such that a cycle of decryption and encryption is not interrupted.

9. A method of claim 8, characterized in that the method further comprises:

- generating initially, via the encoder, the encryption information from at least one initialization value; and

- obfuscating the at least one initialization value stored in its corresponding allocated portion of the data memory or an additional data memory prior to releasing the corresponding allocated portion of the data memory or the additional data memory.

10. A method of claim 8 or 9, characterized in that the additional data memory is different from the data memory used to store given encrypted data (E2, E3, EN or EN+1).

11. A method of any one of claims 8 to 10, characterized in that the unencrypted data (D1) comprises unencrypted data values of at least one protected variable that are to be

utilized during a runtime execution of a program, and the encryption information comprises at least one key to be used to encrypt the unencrypted data values to generate encrypted data values and/or to decrypt the encrypted data values to re-generate the unencrypted data values.

12. A method of any one of claims 8 to 10, characterized in that the unencrypted data (D1) comprises unencrypted data values of at least one protected variable that are to be utilized during a runtime execution of a program, and the encryption information comprises an index of at least one key to be used to encrypt the unencrypted data values to generate encrypted data values and/or to decrypt the encrypted data values to re-generate the unencrypted data values, wherein the method further comprises generating by or accessing from a key store the at least one key using the index.

13. A method of any one of claims 8 to 12, characterized in that the encryption information is generated randomly.

14. A computer program product comprising a non-transitory computer-readable storage medium having computer-readable instructions stored thereon, the computer-readable instructions being executable by a computerized device comprising processing hardware to execute a method as claimed in any one of claims 8 to 13.