(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 111444518 A
(43)申请公布日 2020.07.24

(21)申请号 201910970322.1

(22)申请日 2019.10.12

(30)优先权数据
10-2019-0005855 2019.01.16 KR

(71)申请人 三星电子株式会社
地址 韩国京畿道

(72)发明人 金栽赫 申钟勋 姜智守 金贤镒 李惠秀 崔弘默

(74)专利代理机构 北京市立方律师事务所 11330
代理人 李娜

(51)Int.Cl.
G06F 21/60(2013.01)
G06F 21/71(2013.01)
G06F 21/75(2013.01)

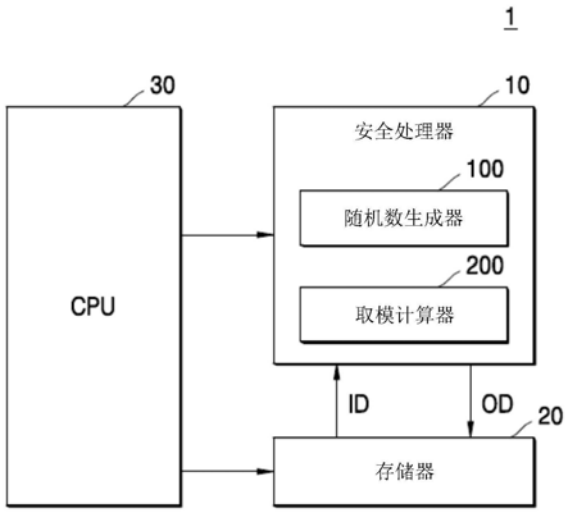
权利要求书4页 说明书15页 附图9页

(54)发明名称

安全处理器及其操作方法、加密或解密数据的方法

(57)摘要

提供了安全处理器及其操作方法、加密或解密数据的方法。所述安全处理器包括:随机数生成器,所述随机数生成器被配置为生成第一随机数;以及取模计算器,所述取模计算器被配置为基于第一输入数据和所述第一随机数生成第一随机操作数,并且通过对所述第一随机操作数进行取余运算生成输出数据,其中,对所述第一输入数据进行所述取余运算的结果值与对所述第一随机操作数进行所述取余运算的结果值相同。



CN 111444518 A

1.一种安全处理器,包括:

随机数生成器,所述随机数生成器被配置为生成第一随机数;以及

取模计算器,所述取模计算器被配置为基于第一输入数据和所述第一随机数生成第一随机操作数,并且通过对所述第一随机操作数进行取余运算来生成输出数据,

其中,对于所述第一输入数据的任何值而言,对所述第一输入数据进行所述取余运算的结果值都与对所述第一随机操作数进行所述取余运算的结果值相同。

2.根据权利要求1所述的安全处理器,其中,所述取模计算器包括:

第一随机操作数生成器,所述第一随机操作数生成器被配置为生成所述第一随机操作数;

输出数据生成器,所述输出数据生成器被配置为通过在对第二输入数据和所述第一随机操作数执行算术运算之后执行所述取余运算来生成所述输出数据。

3.根据权利要求2所述的安全处理器,其中,所述第一随机操作数生成器包括:

乘法器,所述乘法器被配置为将模与所述第一随机数相乘,所述模是所述取余运算的被除数;

加法器,所述加法器被配置为通过将所述第一输入数据与所述乘法器的乘法结果相加来生成所述第一随机操作数。

4.根据权利要求3所述的安全处理器,其中,所述乘法器包括:

重编码器,所述重编码器被配置为生成分别与构成所述第一随机数的多个单元随机数相对应的多个控制信号;

单元模生成器,所述单元模生成器被配置为生成与通过基于所述多个控制信号将所述模与所述多个单元随机数相乘而获得的值相对应的多个单元模,

其中,所述加法器被配置为通过将所述多个单元模与所述第一输入数据按数位数相加来生成所述第一随机操作数。

5.根据权利要求4所述的安全处理器,其中,所述重编码器被配置为将所述多个单元随机数中的每一个单元随机数转换为多个重编码值,并且基于所述多个重编码值生成所述多个控制信号。

6.根据权利要求5所述的安全处理器,其中,所述重编码器包括:

重编码表,所述重编码表存储用于基于布斯重编码将所述多个单元随机数转换为所述多个重编码值的转换信息;以及

控制信号表,所述控制信号表存储关于所述多个控制信号分别与所述多个重编码值的对应的对应信息,

其中,所述重编码器被配置为基于所述重编码表将所述多个单元随机数转换为所述多个重编码值,并且基于所述控制信号表向所述单元模生成器输出用于所述多个重编码值中的每一个重编码值的所述多个控制信号,并且

其中,所述单元模生成器被配置为基于所述多个控制信号生成所述多个单元模。

7.根据权利要求4所述的安全处理器,还包括存储器,所述存储器存储包括所述第一输入数据的原始数据,

其中,所述取模计算器被配置为顺序地加载通过将所述原始数据以m比特为单位进行分割而获得的多条分割数据,

其中，m是自然数。

8.根据权利要求7所述的安全处理器，其中，所述加法器被配置为在对所述多条分割数据中的第一分割数据执行第一运算之前对所述多条分割数据中的第二分割数据执行第二运算，

其中，所述第一随机操作数生成器还包括第一寄存器，所述第一寄存器用于存储在所述第二运算中超出m比特的向上舍入值，并且

其中，所述加法器被配置为通过将从所述第一寄存器加载的向上舍入值、所述多个单元模和所述第一分割数据按数位数相加来生成所述第一随机操作数。

9.根据权利要求2所述的安全处理器，其中，

所述随机数生成器被配置为还生成第二随机数，并且

其中，所述取模计算器还包括第二随机操作数生成器，所述第二随机操作数生成器被配置为基于所述第二输入数据和所述第二随机数生成第二随机操作数，并且

其中，所述输出数据生成器被配置为通过在对所述第一随机操作数和所述第二随机操作数执行所述算术运算之后执行所述取余运算来生成所述输出数据。

10.根据权利要求2所述的安全处理器，其中，所述随机数生成器还被配置为生成与所述第一随机数不同的第三随机数，并且

所述第一随机操作数生成器还被配置为在对所述第二输入数据的运算过程中，基于所述第一输入数据和所述第三随机数生成第三随机操作数。

11.根据权利要求10所述的安全处理器，其中，

所述第二输入数据包括第一子输入数据和第二子输入数据，并且

其中，所述输出数据生成器被配置为：

通过在对所述第一随机操作数和所述第一子输入数据执行所述算术运算之后执行所述取余运算来生成第一子输出数据，

通过在对所述第三随机操作数和所述第二子输入数据执行所述算术运算之后执行所述取余运算来生成第二子输出数据，以及

通过在将所述第一子输出数据和所述第二子输出数据相加之后执行所述取余运算来生成所述输出数据。

12.根据权利要求10所述的安全处理器，其中，所述第一随机操作数生成器还包括第二寄存器，所述第二寄存器存储从所述随机数生成器接收到的至少一个随机数，并且

其中，所述第一随机操作数生成器被配置为在向所述输出数据生成器输出所述第一随机操作数之后向所述随机数生成器输出随机数请求信号，并且

其中，所述随机数生成器被配置为响应于所述随机数请求信号向第一寄存器输出所述第三随机数。

13.一种安全处理器，所述安全处理器包括：

随机数生成器，所述随机数生成器被配置为生成第一随机数；

取模计算器，所述取模计算器被配置为：通过将模与所述第一随机数相乘，将所述相乘的结果与第一输入数据相加来生成第一随机操作数，然后执行将所述第一随机操作数除以所述模的取余运算，来生成输出数据。

14.根据权利要求13所述的安全处理器，其中，所述取模计算器被配置为对第二输入数

据和所述第一随机操作数执行算术运算，以及输出余数作为所述输出数据，所述余数是通过将所述算术运算的结果除以所述模得到的。

15.根据权利要求14所述的安全处理器，其中，

所述随机数生成器还被配置为生成第二随机数，并且

所述取模计算器被配置为：将所述模与所述第二随机数相乘，通过将所述相乘的结果与所述第二输入数据相加来生成第二随机操作数，对所述第一随机操作数和所述第二随机操作数执行所述算术运算，以及生成余数作为所述输出数据，所述余数是将所述算术运算的结果除以所述模而生成的。

16.根据权利要求14所述的安全处理器，其中，

所述随机数生成器被配置为还生成与所述第一随机数不同的第三随机数，并且

所述取模计算器被配置为还通过在对所述第二输入数据的运算过程中将所述模与所述第三随机数相乘，然后将所述相乘的结果与所述第一输入数据相加来生成第三随机操作数。

17.根据权利要求16所述的安全处理器，其中，

所述第二输入数据包括第一子输入数据和第二子输入数据，并且

其中，所述取模计算器被配置为：对所述第一随机操作数和所述第一子输入数据执行第一算术运算，通过将所述第一算术运算的结果值除以所述模生成第一子输出数据，对所述第三随机操作数和所述第二子输入数据执行第二算术运算，通过将所述第二算术运算的结果值除以所述模生成第二子输出数据，以及基于所述第一子输出数据和所述第二子输出数据生成所述输出数据。

18.根据权利要求13所述的安全处理器，其中，所述取模计算器被配置为：

基于对所述第一随机数进行布斯重编码来生成与所述第一随机数对应的多个控制信号，以及

基于所述多个控制信号生成与将所述第一随机数乘以所述模的结果值相对应的多个单元模。

19.根据权利要求18所述的安全处理器，其中，所述取模计算器被配置为通过将所述多个单元模与所述第一输入数据按数位数相加来生成所述第一随机操作数。

20.一种安全处理器的操作方法，所述操作方法包括：

生成第一随机数；

基于模、所述第一随机数和第一输入数据生成第一随机操作数；以及

基于所述第一随机操作数生成输出数据，

其中，对于所述第一输入数据的任何值而言，所述第一随机操作数除以所述模的余数都与所述第一输入数据除以所述模的余数相同。

21.根据权利要求20所述的安全处理器的操作方法，其中，所述第一随机操作数的生成包括：

将所述模与所述第一随机数相乘；以及，

将所述相乘的结果值与所述第一输入数据相加，

其中，作为所述相加的结果而生成的数据是所述第一随机操作数。

22.根据权利要求20所述的安全处理器的操作方法，其中，所述输出数据的生成包括：

对第二输入数据和所述第一随机操作数执行算术运算;以及,

生成余数作为所述输出数据,所述余数是通过将执行所述算术运算的结果值除以所述模而得到的。

23.根据权利要求22所述的安全处理器的操作方法,所述方法还包括:

生成第二随机数;以及,

基于所述模、所述第二随机数和所述第二输入数据生成第二随机操作数,

其中,所述输出数据的生成包括:

对所述第一随机操作数和所述第二随机操作数执行所述算术运算;以及,

生成余数作为所述输出数据,所述余数是通过将执行所述算术运算的结果值除以所述模而得到的。

24.根据权利要求22所述的安全处理器的操作方法,所述方法还包括:

生成第三随机数;以及

在对所述第二输入数据的运算中,基于所述模、所述第三随机数和所述第一输入数据,生成第三随机操作数。

25.一种加密或解密数据的方法,所述方法包括:

生成随机数;

将所述随机数与模相乘;

通过将所述相乘的结果与输入数据相加来生成随机操作数;

通过使用所述模对所述随机操作数执行取余运算来生成输出数据;以及

基于所述输出数据执行密码操作。

安全处理器及其操作方法、加密或解密数据的方法

[0001] 相关申请的交叉引用

[0002] 本申请要求于2019年1月16日在韩国知识产权局提交的韩国专利申请No.10-2019-0005855的优先权，该韩国申请的全部公开内容以引用的方式合并于本申请中。

技术领域

[0003] 本发明构思涉及一种安全处理器，更具体地，涉及一种能够有效地保护计算机系统免受侧信道攻击(SCA(side channel attack))的安全处理器及安全处理器的操作方法。

背景技术

[0004] 安全处理器可以通过使用要求安全性的信息(例如密钥)执行诸如密码操作的安全算法。基于安全处理器实施密码操作的机密信息的外泄可以被称为侧信道，并且使用侧信道的攻击方法可以被称为SCA。安全处理器可以采用针对SCA的对策来避免危及密码操作。

[0005] 模板攻击是SCA的一个示例。在模板攻击中，攻击者可以使用与攻击目标类似的设备来创建“模板”。攻击者可以访问他们的目标设备的副本上的实际上无限次的输入和攻击操作。因此，攻击者可以通过针对设备的副本上处理的大量输入记录侧信道信息，来创建侧信道信息与目标设备上的机密信息如何相关的模型。例如，攻击者可以创建在执行密码操作时目标设备上的功率大小与输入如何对应的模型。

[0006] 针对SCA的防御技术(包括屏蔽或隐藏侧信道信息)可能影响安全处理器的性能。例如，实施防御技术可能会使安全处理器的电路面积或平均功耗增加。此外，这些防御技术可能无法针对诸如模板攻击和功率分析攻击的某些SCA提供充分的保护。

发明内容

[0007] 本发明构思提供了一种安全处理器及其操作方法，所述安全处理器能够有效地防止侧信道攻击(SCA)，同时使所增加的功耗或性能开销最小化。

[0008] 根据本发明构思的一个方面，提供了一种安全处理器，包括：随机数生成器，所述随机数生成器被配置为生成第一随机数；以及取模计算器，所述取模计算器被配置为基于第一输入数据和所述第一随机数生成第一随机操作数，并且通过对所述第一随机操作数进行取余运算来生成输出数据，其中，对于所述第一输入数据的任何值而言(即，对于所述安全处理器要处理的所述第一输入数据的任何值而言)，对所述第一输入数据进行所述取余运算的结果值都与对所述第一随机操作数进行所述取余运算的结果值相同。

[0009] 根据本发明构思的另一方面，提供了一种安全处理器，所述安全处理器包括：随机数生成器，所述随机数生成器被配置为生成第一随机数；取模计算器，所述取模计算器被配置为：通过将模与所述第一随机数相乘，将所述相乘的结果与第一输入数据相加来生成第一随机操作数，然后执行将所述第一随机操作数除以所述模的取余运算，来生成输出数据。

[0010] 根据本发明构思的一个方面，提供了一种安全处理器的操作方法，所述操作方法

包括:生成第一随机数;基于模、所述第一随机数和第一输入数据生成第一随机操作数;以及基于所述第一随机操作数生成输出数据,其中,对于所述第一输入数据的任何值而言,所述第一随机操作数除以所述模的余数都与所述第一输入数据除以所述模的余数相同。

[0011] 描述了一种加密或解密数据的方法。所述方法包括:生成随机数;将所述随机数与模相乘;通过将所述相乘的结果与输入数据相加来生成随机操作数;通过使用所述模对所述随机操作数执行取余运算来生成输出数据;以及基于所述输出数据执行密码操作(例如,加密数据或解密数据)。

附图说明

[0012] 从下文结合附图的详细描述将会更加清楚地理解本发明构思的实施例,在附图中:

[0013] 图1是示出了根据示例实施例的电子系统的框图;

[0014] 图2是示出了根据示例实施例的安全处理器的框图;

[0015] 图3是示出了根据示例实施例的随机操作数生成器的框图;

[0016] 图4是示出了根据示例实施例的安全处理器的框图;

[0017] 图5和图6是示出了根据示例实施例的算术处理器的操作的示图;

[0018] 图7是示出了根据示例实施例的重编码器的示图;

[0019] 图8示出了根据示例实施例的重编码表;

[0020] 图9示出了根据示例实施例的控制信号表;

[0021] 图10是示出了根据示例实施例的局部乘法器的电路图;

[0022] 图11是示出了根据示例实施例的安全处理器的框图;

[0023] 图12是示出了根据示例实施例的安全处理器的框图;

[0024] 图13是示出了根据示例实施例的安全处理器的操作的示图;

[0025] 图14是示出了根据示例实施例的安全处理器的框图;以及

[0026] 图15是示出了根据示例实施例的应用处理器的框图。

具体实施方式

[0027] 图1是示出了根据示例实施例的电子系统1的框图。

[0028] 参考图1,电子系统1可以包括安全处理器10、存储器20和中央处理单元(CPU)30。电子系统1可以对应于包括例如膝上型计算机、移动电话、智能电话、平板个人计算机(PC)和个人数字助理(PDA)等各种类型的系统。

[0029] CPU 30可以输出用于控制安全处理器10和存储器20的各种控制信号。在另一实施例中,应用处理器(AP)可以执行CPU 30的功能。

[0030] 存储器20可以在CPU 30或安全处理器10的控制下存储数据。在本发明构思的实施例中,存储器20可以记录从外部源接收的输入数据ID,在CPU 30的控制下将输入数据ID提供给安全处理器10,接收来自安全处理器10的输出数据OD,并记录接收的输出数据OD,其中输出数据OD是对输入数据ID执行安全操作的结果。

[0031] 电子系统1还可以包括安全处理器10(可以与CPU 30分离),该安全处理器10能够实现与安全操作有关的快速算术处理。安全处理器10可以使用机密信息执行操作。安全处

理器10还可以被称为安全计算器。在一个实施例中，安全处理器10可以使用公钥基础设施(PKI)中的个人密钥、私钥或两者来执行加密或解密操作。

[0032] 安全处理器10可以执行与加密或解密操作有关的各种操作。例如，安全处理器10可以执行与加密或解密数据有关的全部操作。或者，安全处理器10可以仅执行与全部加密或解密操作有关的多个操作中的一些操作。

[0033] 根据本发明构思的实施例，被安全处理器10的加密或解密操作处理的数据可以被称为输入数据ID，作为加密或解密操作的结果而生成的数据可以被称为输出数据OD。在一个实施例中，输入数据ID可以表示PKI中的私钥或公钥，输出数据OD可以表示至少部分地基于私钥、公钥或两者被加密或解密的数据。

[0034] 安全处理器10中的加密或解密操作可以包括一个或更多个乘法运算。例如，当安全处理器10执行公钥算法时，安全处理器10可以对相对较大的数执行算术运算(加法、减法、乘法、取余运算和取模运算等)。在一些情况下，当操作数的大小大于某个阈值(例如，在Rivest Shamir Adleman(RSA)算法的情况下至少为1024比特)时，操作可以被识别为是安全的。

[0035] 对于使用大操作数的操作，安全处理器10可以实现数字串行乘法(Digital-serial Multiplication)。作为示例，将用于执行一般串行乘法(Serial Multiplication)的算法1描述如下：

[0036]

Algorithm 1: Serial multiplication

Inputs: Positive integers A and B, where $B = \sum_{i=0}^{n-1} b_i 2^i$, n is the operand size, and $b_i = 0$ or **1**

Output: The result of the multiplication: $C \leftarrow A * B$

1. $C \leftarrow 0$
2. For i from $n - 1$ downto 0 do
   - A. $T \leftarrow C * 2$
   - B. $T \leftarrow T + b_i * A$
   - C. $C \leftarrow T$
3. Return C

[0037]

算法 1：串行乘法

输入：正整数 A 和 B，其中$B = \sum_{i=0}^{n-1} b_i 2^i$，n 是操作数大小，并且 b_i=0 或 1

输出：乘法的结果：$C \leftarrow A * B$

1. $C \leftarrow 0$
2. i 从 n-1 降到 0，执行
   A. $T \leftarrow C * 2$
   B. $T \leftarrow T + b_i * A$
   C. $C \leftarrow T$
3. 返回 C

[0038] 算法1

[0039] 作为另一示例,将用于执行数字串行乘法的算法2描述如下：

[0040]

Algorithm 2: Digit-serial Multiplication

Inputs: Positive integers A and B, where $B = \sum_{i=0}^{k-1} b_i 2^{di}$, n is the operand size, d is the digit size, $k = \lceil n/d \rceil$, and $b_i = [0,1,2,\ldots,2^d - 1]$

Output: The result of the multiplication: $C \leftarrow A * B$

1. $C \leftarrow 0$
2. For i from $k - 1$ downto 0 do
   A. $T \leftarrow C * 2^d$
   B. $T \leftarrow T + b_i * A$
   C. $C \leftarrow T$
3. Return C

[0041]

算法 2：数字串行乘法

输入：正整数 A 和 B，其中$B = \sum_{i=0}^{n-1} b_i 2^{di}$，n 是操作数大小，d 是数位大小，k=[n/d]，并且 b_i=[0,1,2,…,2^d-1]

输出：乘法的结果：$C \leftarrow A * B$

1. $C \leftarrow 0$

2. i 从 k-1 降到 0，执行

A. $T \leftarrow C * 2^d$

B. $T \leftarrow T + b_i * A$

C. $C \leftarrow T$

3. 返回 C

[0042] 算法2

[0043] 参考上述算法1和算法2,数字串行乘法是一般串行乘法的特殊情况,它能够一次对乘数B的几个比特b_i进行运算。

[0044] 在执行如上所述的乘法运算时,安全处理器10消耗的功率会受到运算值的影响。因此,当频繁执行类似操作时,可以使用诸如模板攻击的SCA来识别输入数据值(例如,A或B)。另外地或作为另一种选择,还可以分析安全处理器10的诸如密钥的机密信息。

[0045] 可以采用隐藏技术和屏蔽技术作为针对这种类型的SCA的对策。隐藏技术可以指通过减少侧信道信号(或通过增加噪声)来减少在处理不同输入时的功耗变化的方法。然而,隐藏技术可能导致电路面积、平均功耗或两者的显著增加(例如,增加超过两倍)。另外,如果仅在时钟信号处于特定状态(例如,低状态)时执行实际运算,则会降低整体性能。

[0046] 屏蔽技术(masking technique)可以指通过在加密操作之前执行屏蔽操作来使输入随机化。然后可以在加密操作之后执行解屏蔽操作以抵消屏蔽操作(得到在数学上等效的操作)。然而,当乘法运算中的输入数据值(即,A或B)对应于“0”时,屏蔽效果会丧失,因此安全处理器10的信息可能依然存在信息泄露的风险。

[0047] 因此,根据本公开的实施例,安全处理器10可以通过基于随机数改变作为SCA的目标的输入数据ID,来向输入数据ID提供随机特性。这可以使安全处理器10能够在将输入数据ID作为目标的SCA期间使泄漏信息的风险最小化。也即是,可以随机地改变对实际输入数据ID的运算期间的功耗,因此可以消除或减少抗模板攻击的脆弱性。

[0048] 为此,安全处理器10可以包括随机数生成器100和取模计算器200。随机数生成器100可以生成随机数。例如,随机数生成器100可以基于由用户的运动、电阻器的热噪声、半导体的p-n结的短噪声和光子产生的短噪声、辐射波等产生的熵信号,来产生随机数。在另一示例中,随机数生成器100可以基于在亚稳态下随机变化的熵信号来生成随机数。

[0049] 取模计算器200可以基于从存储器20接收的输入数据ID和从随机数生成器100接收的随机数生成随机操作数,并且可以通过对随机操作数执行取余运算来输出输出数据OD。取余运算可以是将通过除数除以被除数而得到的余数输出的运算,也可以称为取模运

算。

[0050] 因此,电子系统1可以通过如下方式执行密码操作:使用随机数生成器100生成随机数;将随机数乘以模,通过将乘积的结果与输入数据ID相加来产生随机操作数,通过在取模计算器200处使用模对随机操作数执行取余运算来产生输出数据OD;以及基于输出数据OD执行密码操作。

[0051] 根据本发明构思的技术思想,取模计算器200可以对基于随机数和输入数据ID生成的随机操作数执行取余运算,该随机操作数可以与输入数据ID本身不同。由于随机操作数的特性根据随机数变化,因此安全处理器10可以具有对SCA的很强的免疫力。例如,使用随机操作数可以使输入数据ID与从安全处理器10泄漏的信息(例如,功率使用信息)之间的关系模糊。

[0052] 图2是示出了根据示例实施例的安全处理器的框图。省略了先前参考图1给出的描述。

[0053] 参考图2,安全处理器10可以包括随机数生成器100和取模计算器200,并且取模计算器200可以包括随机操作数生成器210和输出数据生成器220。

[0054] 随机操作数生成器210可以从随机数生成器100接收随机数RN,从存储器(图1中的20)接收第一输入数据ID1。随机操作数生成器210可以基于随机数RN和第一输入数据ID1生成随机操作数RO。在一个实施例中,随机操作数生成器210可以通过将随机数RN和第一输入数据ID1输入到特定公式中来生成随机操作数RO。在示例中,随机操作数生成器210可以通过如下方式生成随机操作数RO:将模M与随机数RN相乘,然后通过将相乘的结果与第一输入数据ID1相加来生成随机操作数RO(其中RO=ID1+RN*M)。模M可以是输出数据生成器220为生成输出数据OD而执行的取余运算的被除数。

[0055] 输出数据生成器220可以从随机操作数生成器210接收随机操作数RO,从存储器(图1中的20)接收第二输入数据ID2。输出数据生成器220可以对随机操作数RO和第二输入数据ID2执行算术运算(例如,加法、减法、乘法和除法),并且可以通过执行将模M用作被除数的取余运算来生成输出数据OD。

[0056] 根据取余运算(mod)的性质,诸如A、B、r和模M的任意数以及任意算术运算△(例如,加法、减法、乘法和除法中的任何一种),可以满足下面的公式1和公式2。

[0057] [公式1]

[0058] $A \bmod M = (A+rM) \bmod M$

[0059] [公式2]

[0060] $(A \triangle B) \bmod M = \{(A+rM) \triangle B\} \bmod M$

[0061] $= \{A \triangle (B+rM)\} \bmod M$

[0062] 根据上述公式1和公式2,第一输入数据ID1、第二输入数据ID2和随机操作数RO可以满足公式3和公式4。

[0063] [公式3]

[0064] $ID1 \bmod M = (ID1+RN*M) \bmod M$

[0065] $= RO \bmod M$

[0066] [公式4]

[0067] $(ID1 \triangle ID2) \bmod M = \{(ID1+RN*M) \triangle ID2\} \bmod M$

[0068] ＝(RO△ID2)mod M

[0069] 根据本发明构思的技术思想的安全处理器10可以基于随机数RN而不是第一输入数据ID1来对随机操作数RO执行取余运算。根据取余运算的性质，对随机操作数RO进行的取余运算的结果可以与对第一输入数据ID1进行的取余运算的结果相同，并且安全处理器10可以通过对添加了随机特性的随机操作数RO执行取余运算来有效地降低SCA的风险。

[0070] 也即是，因为取余运算是对随机化后的输入RO△ID2进行运算的，而不是对输入本身(即，ID1△ID2)进行运算的，所以安全处理器10可以减小侧信道信息泄漏的可能性。但是，运算本身和取余运算的输出保持不变。因此，安全处理器10可以在基本上不增加使用功率或芯片面积的情况下产生期望的输出并降低SCA的风险。

[0071] 图3是示出了根据示例实施例的随机操作数生成器的框图。

[0072] 参考图3，随机操作数生成器210可以包括乘法器211和加法器212。乘法器211可以接收随机数RN和模M，并且可以对随机数RN和模M执行乘法。在示例中，乘法器211可以从安全处理器(例如，图1中的安全处理器10)内的寄存器接收模M。

[0073] 在实施例中，乘法器211可以将随机数RN分为多个单元随机数，并且可以通过将模M与多个单元随机数中的每一个单元随机数相乘来生成多个单元模UM。乘法器211可以向加法器212输出所生成的多个单元模UM。

[0074] 根据实施例，乘法器211可以将随机数RN分为具有小单元大小(例如，2比特)的多个单元随机数，可以执行对多个单元随机数中的每一个单元随机数和模M进行乘法运算的多个局部乘法运算(partial multiplication operation)，并且因此可以有效地对比特数多的随机数和模M执行乘法运算。加法器212可以将多个单元模UM与第一输入数据ID1按数位数相加来生成随机操作数RO。

[0075] 图4是示出了根据示例实施例的安全处理器的框图。

[0076] 参考图4，安全处理器10可以包括随机数寄存器213、乘法器211、加法器212、随机操作数寄存器216和内部存储器300，并且乘法器211可以包括重编码器214和局部乘法器215。

[0077] 随机数寄存器213可以从随机数生成器(图1中的100)接收随机数RN，并且向乘法器211输出具有特定比特数(例如，t比特)的随机数RN。

[0078] 重编码器214可以接收随机数RN并对接收到的随机数RN执行重编码操作。在示例中，重编码操作可以对应于布斯(Booth)重编码操作，布斯(Booth)重编码操作适用于用来实现随机数RN和模M的数字串行算法的逻辑元件。

[0079] 可以执行多个局部乘法运算，每一个局部乘法运算是随机数RN与模M相乘的一部分，并且可以执行与每个局部乘法运算相对应的一个重编码操作。重编码操作可以包括用在数学上等于原始值的不同值替换随机数的值以有效地实现乘法。当执行重编码操作时，可以减少局部乘法运算的次数。典型的重编码操作可以包括布斯重编码操作。

[0080] 在一个实施例中，重编码器214可以包括存储有用于将多个单元随机数RN中的每一个单元随机数转换为多个重编码值的转换信息的重编码表，并且可以基于重编码表从多个单元随机数中的每一个单元随机数生成多个重编码值。下面参考图7详细描述重编码表。

[0081] 在实施例中，重编码器214可以包括存储有关于多个控制信号SEL与多个重编码值中的每一个重编码值的对应的对应信息的控制信号表，并且可以基于控制信号表向局部乘

法器215输出用于多个重编码值中的每一个重编码值的多个控制信号SEL。下面参考图8详细描述控制信号表。

[0082] 局部乘法器215可以接收m比特的模M和多个控制信号SEL，并基于多个控制信号SEL生成多个单元模UM。多个单元模UM可以分别对应于通过将多个单元随机数乘以模M而获得的值。

[0083] 内部存储器300可以存储从安全处理器10的外部(例如，从图1中的存储器20)接收的原始数据OrgD，并且可以向加法器212输出分割数据SD(原始数据OrgD以m比特为单位进行分割后的数据)。为此，内部存储器300可以被实现为至少一个存储设备，并且可以包括例如易失性存储器和非易失性存储器中的至少一种。

[0084] 非易失性存储器可以包括闪速存储器、随机存取存储器(RAM)、相变RAM(PRAM)、磁RAM(MRAM)、电阻RAM(RRAM)和铁电RAM(FRAM)等，易失性存储器可以包括动态RAM(DRAM)、静态RAM(SRAM)、同步DRAM(SDRAM)、锁存器、触发器和寄存器等。

[0085] 加法器212可以接收多个单元模UM和分割数据SD，并且还可以从随机操作数寄存器21接收向上舍入值(roundup value)Cr。向上舍入值Cr可以表示这样的数据：该数据对应于在针对前一分割数据SD的计算中超出作为分割数据SD的比特数的m比特的部分。加法器212可以通过将多个单元模UM、分割数据SD和向上舍入值Cr按数位数相加来产生(m+a)比特的随机操作数RO。加法器212可以将所生成的随机操作数RO存储在随机操作数寄存器216中。

[0086] 随机操作数寄存器216可以存储随机操作数RO，并且当完成对所有原始数据OrgD的操作时，可以输出所存储的随机操作数RO。在图4中，随机操作数寄存器216、随机数寄存器213和内部存储器300被示出为彼此分离。然而，本实施例不限于此。随机操作数寄存器216、随机数寄存器213和内部存储器300中的两者以上可以实现为单个存储元件。此外，在图4中，示出了加法器212从内部存储器300接收分割数据SD的示例。然而，这仅是示例，加法器212可以从安全处理器10外部的存储器(例如，图1中的20)接收分割数据SD。

[0087] 图5和图6是分别示出了根据示例实施例的算术处理器的操作的示图。

[0088] 参考图4和图5，乘法器221可以接收随机数RN(例如，4比特的随机数RN)并且将随机数RN分成第一单元随机数URN1和第二单元随机数URN2，第一单元随机数URN1和第二单元随机数URN2均以两比特为单位。乘法器221可以通过将第一单元随机数URN1与8比特的模M相乘来生成10比特的第一单元模UM1。乘法器221可以通过将第二单元随机数URN2与模M(例如，具有8比特的模)相乘来生成10比特的第二单元模UM2。

[0089] 根据本发明构思的实施例，乘法器221可以基于重编码表计算用于随机数RN的多个重编码值，并且可以基于控制信号表通过使用与多个重编码值中的每一个重编码值相对应的多个控制信号将模M与第一单元随机数和第二单元随机数相乘。参考图7至图10对此进行详细描述。

[0090] 图5示出了第一单元模UM1和第二单元模UM2都具有10比特的实施例，但是本发明构思不限于此。第一单元模UM1的比特数和第二单元模UM2的比特数可以大于或小于10比特，这取决于第一单元随机数URN1和第二单元随机数URN2与模M相乘的结果。

[0091] 参考图4和图6，加法器212可以从乘法器221接收第一单元模UM1和第二单元模UM2，从内部存储器300接收分割数据SD，从随机操作数寄存器216接收第一向上舍入值Cr1。

加法器212可以将第一单元模块UM1、第二单元模块UM2、分割数据SD与第一向上舍入值Cr1按数位数相加。在一个示例中，第二单元模UM2是通过将第二单元随机数URN2与模M相乘而获得的值，第二单元随机数URN2的数位的位置比第一单元随机数URN1的数位的位置高两位。因此，为了获得所需的总数位数(total number of digits)，可以通过将第二单元随机数URN2偏移两位来执行相加。

[0092] 作为相加的结果，加法器212可以生成12位的数据。所生成的数据中的第一位o0至第八位o7(即，8比特并且对应于分割数据SD)可以作为计算完成的随机操作数RO而被存储在随机操作数寄存器216中。在对下一个分割数据SD的加法运算中，第九位o8至第十二位o11可以用作第二向上舍入值Cr2。

[0093] 图7是示出了根据示例实施例的重编码器的示图。

[0094] 参考图4和图7，重编码器可以包括用于存储重编码器输入与重编码值之间的转换信息的重编码表。转换信息可以包括分别与多个重编码器输入的模式相对应的多个重编码值。重编码器输入Reco_In可以包括单元随机数的比特b_i和参考比特bref。重编码器输入Reco_In可以根据比特(b_i和bref)具有与各种模式中的一种模式相对应的形式。例如，重编码器输入Reco_In可以对应于第一模式Pat(1)至第八模式Pat(8)中的任何一种模式。在一个示例中，单元随机数包括2比特，并且参考比特bref是1比特。

[0095] 根据存储在重编码表中的转换信息，可以将重编码器输入Reco_In映射或转换为与其对应的重编码值。在示例中，与具有第一模式Pat(1)的重编码器输入Reco_In相对应的第一重编码值Val(1)被存储在转换信息中。在该示例中，当单元随机数的比特b_i和参考比特bref具有第一模式Pat(1)时，重编码器可以输出第一重编码值Val(1)。

[0096] 附加地或作为另一种选择，可以基于上面的转换运算生成重编码值，并且可以以控制信号sel[0:k]的形式输出重编码值。在一个示例中，当随机数为两比特时，控制信号sel[0:k]可以包括四比特。

[0097] 根据本发明构思的实施例，当应用上述布斯重编码方法时，重编码输出可以采用来自集合{-2,-1,0,1,2}中的值。因此，对应于具有任何一种模式的重编码器输入的重编码值可以与对应于具有另一模式的重编码器输入的重编码值相同。也即是，可以将多种重编码器输入模式映射到相同的重编码值。

[0098] 另外，基于控制信号sel[0:k]，局部乘法器215可以生成局部乘法运算中随机数RN与{-2,-1,0,1,2}(例如，局部乘法系数)中的任何一个值相乘的结果。因此，在一个局部乘法运算中，局部乘法器215可以接收模M和控制信号sel[0:k]作为输入并通过局部乘法运算过程生成{-2M,-M,0,M,2M}中的任何一个值。

[0099] 图8示出了根据示例实施例的重编码表。

[0100] 参考图8，当随机数RN是8比特值“01101001”时，可以如公式5所示转换随机数RN，使得系数具有{-2,-1,0,1,2}中的值。

[0101] [公式5]

[0102] $RN=01101001_{(2)}$

[0103] $=0\cdot 2^7+1\cdot 2^6+1\cdot 2^5+0\cdot 2^4+1\cdot 2^3+0\cdot 2^2+0\cdot 2^1+1\cdot 2^0$

[0104] $=2\cdot 2^6-1\cdot 2^4-2\cdot 2^2+1\cdot 2^0$ (1)

[0105] $=1\cdot 2^6+2\cdot 2^4+2\cdot 2^2+1\cdot 2^0$ (2)

[0106] 也即是,随机数RN可以转换为(1)$2\cdot 2^6-1\cdot 2^4-2\cdot 2^2+1\cdot 2^0$以及(2)$1\cdot 2^6+2\cdot 2^4+2\cdot 2^2+1\cdot 2^0$。上述两个转换值可以具有与原始随机数RN相同的值105。

[0107] 根据上面的转换示例,单元随机数URN的两个比特可以被转换为值{-2,-1,0,1,2}中的任何一个值。因此,8位随机数RN可以被分成四个单元随机数,每个单元随机数具有两个比特。例如,两个最低有效位可以被分为第一单元随机数,接下来的两个低位可以被分为第二单元随机数,再接下来的两个低位可以被分为第三单元随机数,两个最高有效位可以被分为第四单元随机数。

[0108] 另外,当假设首先从随机数RN的低位执行重编码操作时,执行当前重编码操作的单元随机数的高位可以用作下一个重编码操作中的参考比特。例如,当对第二单元随机数执行重编码操作时,第一单元随机数的两个比特中的高位可以对应于参考比特Ref bit。

[0109] 在对包括两个最低有效位的第一单元随机数执行重编码操作的情况下,可以将对应于“0”的比特定义为参考比特Ref bit。因此,假设执行当前重编码操作的两个比特是b_{2i+1}和b_{2i},则当前重编码操作中的参考比特Ref比特可以对应于b_{2i-1},并且下一个重编码操作中的参考比特(Next Ref bit)可以对应于b_{2i+1}。

[0110] 在随机数RN是“$01101001_{(2)}$”的示例中,第一单元随机数可以是“01”(来自第一位和第二位),并且参考比特可以是“0”。如图8的重编码表中所示,包括第一单元随机数的比特和参考比特Ref的重编码器输入可以对应于“010”,并且与该重编码器输入对应的重编码值Recoding Value可以为“1”。

[0111] 包括接下来的两个低位的第二单元随机数可以是“10”(来自第三位和第四位),并且参考比特Ref bit可以是来自前一个第一单元随机数的高位“0”。因此,包括第二单元随机数的比特和参考比特Ref bit的重编码器输入可以对应于“100”,并且与该重编码器输入对应的重编码值可以为“-2”。

[0112] 包括接下来的两个低位的第三单元随机数也可以是“10”(来自第五位和第六位),并且参考比特Ref比特可以是来自前一个第二单元随机数的高位“1”。因此,包括第三单元随机数的比特和参考比特Ref bit的重编码器输入可以对应于“101”,并且与该重编码器输入对应的重编码值可以为“-1”。

[0113] 包括接下来的两个低位的第四单元随机数可以是“01”(来自第七位和第八位),并且参考比特Ref bit可以是来自前一个第三单元随机数的高位“1”。因此,包括第四单元随机数的比特和参考比特Ref bit的重编码器输入可以对应于“011”,并且与该重编码器输入对应的重编码值可以为“-2”。

[0114] 换句话说,重编码器可以接收“$01101001_{(2)}$”作为随机数RN并生成“2”、“-1”、“-2”和“1”作为重编码值。

[0115] 在一些示例中,重编码表是预先生成并且存储在安全处理器中的,但是实施例不限于此。在一个示例中,可以通过计算特定公式获得重编码表中示出的与重编码器输入相对应的重编码值,并且可以在安全处理器中提供用于计算特定公式的配置,以通过运算过程生成重编码值。

[0116] 图9示出了根据示例实施例的控制信号表。

[0117] 参考图8和图9,当图9的控制信号表中所示的重编码值对应于“-2”时,第一控制信号sel0至第四控制信号sel3可以具有值“1010”,当重编码值对应于“-1”时,与其对应的第

一控制信号sel0至第四控制信号sel3可以具有值“1001”。另外，当重编码值对应于“0”时，第一控制信号sel0至第四控制信号sel3可以具有值“0000”。当重编码值对应于“1”时，第一控制信号sel0至第四控制信号sel3可以具有值“0101”。当重编码值对应于“2”时，第一控制信号sel0至第四控制信号sel3可以具有值“0110”。上述第一控制信号sel0至第四控制信号sel3可以用作用于在下面描述的局部乘法运算中选择与模相乘的系数的信号。

[0118] 根据本发明构思的实施例，重编码器可以向局部乘法器提供与参考图8描述的重编码值(例如，“2”，“-1”，“-2”和“1”)对应的第一控制信号sel0至第四控制信号sel3。在示例中，重编码器可以基于“1”(即，对应于第一单元随机数的重编码值)将“0101”作为第一控制信号sel0至第四控制信号sel3输出到局部乘法器。重编码器可以基于“-2”(即，对应于第二单元随机数的重编码值)将“1010”作为第一控制信号sel0至第四控制信号sel3输出到局部乘法器。

[0119] 图10是示出了根据示例实施例的局部乘法器的电路图。

[0120] 参考图10，可以向局部乘法器225提供比特的模M(M[0]至M[n-1])和根据上述随机数的重编码结果的第一控制信号sel0至第四控制信号sel3。尽管在图10中示出了构成一个局部乘法器225的逻辑元件，但是可以一起执行两个以上的局部乘法。因此，安全处理器10可以包括两个以上的局部乘法器225。局部乘法器225可以生成作为n比特的模M(M[0]至M[n-1])与2比特的单元随机数相乘的结果的(n+1)比特的单元模UMi[n:0]和1比特的符号数据UMi_neg。

[0121] 对于模M，(n+1)比特的单元模UMi[n:0]与1比特的符号数据UMi_neg的局部乘法结果值可以是{-2M,-M,0,M,2M}中的任何一个。1比特的符号数据UMi_neg可以对应于第一控制信号sel0。

[0122] 可以使用多个逻辑元件来实现局部乘法器225。在图10中，示出了局部乘法器225包括多个反相器410、多个AND门和多个OR门的示例。例如，除了n比特的模M(M[0]至M[n-1])之外，还可以在n比特的模M(M[0]至M[n-1])的最低有效位(即，M[0])的右侧进一步输入零值。另外，可以进一步在n比特的模M(M[0]至M[n-1])的最高有效位(即，M[n-1])的左侧输入零值。

[0123] 局部乘法器225可以包括与n比特的模M(M[0]至M[n-1])和上述的两个零值对应的多个(例如，n+2)个反相器225_1。另外，局部乘法器225的第一级还可以包括第一AND门块225_2和第一OR门块225_3。第一AND门块225_2可以包括多个AND门(AND1_11、AND1_12至AND1_(n+2)1、AND1(n+2)2)。作为示例，可以对与n比特的模M(M[0]至M[n-1])的每个比特以及两个零值中的每一个相对应地布置两个AND门。以n比特的模M(M[0]至M[n-1])的最低有效位M[0]为例，第一AND门AND1_21可以接收第二选择信号sel1和最低有效位M[0]作为输入，第二AND门AND1_22可以接收第一选择信号sel0和最低有效位M[0]的反相值作为输入。

[0124] 此外，第一OR门块225_3可以包括第一OR门OR1_1至第(n+2)OR门OR1_(n+2)。以n比特的模M(M[0]至M[n-1])的最低有效位M[0]为例，第一AND门AND1_21的输出和第二AND门AND1_22的输出可以被提供为第二OR门OR1_2的输入。

[0125] 局部乘法器225的第二级还可以包括第二AND门块225_4和第二OR门块225_5。作为示例，第二AND门块225_4可以包括多个AND门(AND2_11、AND2_12至AND2_n1、AND2_n2)。以n比特的模M(M[0]至M[n-1])的两个最低有效位M[0]和M[1]为例，第一AND门AND2_21可以接

收第四选择信号sel3和第一OR门块225_3的第三OR门OR1_3的输出，第二AND门AND2_22可以接收第三选择信号sel2和第一OR门块225_3的第二OR门OR1_2的输出。

[0126] 第二OR门块225_5可以包括第一OR门OR2_1至第n OR门OR2_n。以n比特的模M(M[0]至M[n-1])的最低有效位M[0]为例，可以将第一AND门AND2_21的输出和第二AND门AND2_22的输出提供为第二OR门OR2_2的输入。对包括在局部乘法器225中的其他逻辑元件的详细连接关系可以被实现为如附图中所示的那样，省略其详细描述。

[0127] 在图10所示的逻辑元件中，上述重编码结果可以通过第一控制信号sel0至第四控制信号sel3反映在局部乘法运算中。例如，第一控制信号sel0和第二控制信号sel1可以确定正/负号，第三控制信号sel2和第四控制信号sel3可以确定局部乘法的系数对应于“1”或“2”。

[0128] 当局部乘法结果为负时，符号数据UMi_neg可以用于生成二进制补码。例如，-2M可以被计算为$-2M=\sim(M<<1)+1$。在该等式中，(M<<1)可以对应于乘以2，并且可以使用按位非(“$\sim$”)和“+1”来生成二进制补码。

[0129] 根据本发明构思的实施例的局部乘法器可以生成单元模(UMi[n:[0]和符号数据UMi_neg，对应于通过基于第一控制信号sel0至第四控制信号sel3将单元随机数与模M相乘而获得的值，并且可以将所生成的单元模(UMi[n:0]和UMi_neg输出到加法器(例如，图3的加法器212)。

[0130] 图11是示出了根据示例实施例的安全处理器10a。安全处理器10a可以类似于参考图2描述的安全处理器10，但是可以包括多个随机操作数生成器。省略了先前参考图2给出的描述。

[0131] 参考图11，安全处理器10a可以包括随机数生成器100a和取模计算器200a。取模计算器200a可以包括第一随机操作数生成器210a、第二随机操作数生成器230a和输出数据生成器220a。

[0132] 第一随机操作数生成器210a可以从随机数生成器100a接收第一随机数RN1，从存储器(例如，图1中的20)接收第一输入数据ID1。第一随机操作数生成器210a可以基于第一随机数RN1和第一输入数据ID1生成第一随机操作数RO1。在一个示例中，第一随机操作数生成器210a可以通过如下方式生成第一随机操作数RO1：将模M与第一随机数RN1相乘，然后与第一输入数据ID1相加(其中RO1＝ID1+RN1*M)。

[0133] 第二随机操作数生成器230a可以从随机数生成器100a接收第二随机数RN2，从存储器(例如，图1中的20)接收第二输入数据ID2。第二随机操作数生成器230a可以基于第二随机数RN2和第二输入数据ID2生成第二随机操作数RO2。在一个示例中，第二随机操作数生成器230a可以通过如下方式生成第二随机操作数RO2：将模M与第二随机数RN2相乘，然后与第二输入数据ID2相加(其中RO2＝ID2+RN2*M)。

[0134] 输出数据生成器220a可以通过如下方式生成输出数据OD：接收第一随机操作数RO1和第二随机操作数RO2，对第一随机操作数RO1和第二随机操作数RO2执行算术运算(例如，加法、减法、乘法和除法)，以及对算术运算的结果执行取余运算，在该取余运算中模M用作被除数。

[0135] 根据取余运算(mod)的性质，诸如A、B、r和模M的任意数以及任意算术运算△(例如，加法、减法、乘法和除法中的任何一个)可以满足下面的公式6。

[0136] [公式6]

[0137] $(A\triangle B) \bmod M = \{(A+rM)\triangle(B+rM)\} \bmod M$

[0138] 基于上面的公式6建立用于第一输入数据ID1、第二输入数据ID2、第一随机操作数RO1和第二随机操作数RO2的公式7。

[0139] [公式7]

[0140] $(ID1\triangle ID2) \bmod M = \{(ID1+RN1*M)\triangle(ID2+RN2*M)\} \bmod M$

[0141] $= (RO1\triangle RO2) \bmod M$

[0142] 即,根据本发明构思的实施例的安全处理器10a可以对基于第一输入数据ID1和第一随机数RN1计算出的第一随机操作数RO1执行取余运算,对基于第二输入数据ID2和第二随机数RN2计算出的第二随机操作数RO2执行取余运算。与第一输入数据ID1的情况类似,安全处理器10a可以通过使用利用第二随机数RN2生成的第二随机操作数RO2来对第二输入数据ID2执行取余运算。因此,可以提高安全处理器10a的安全性(例如,通过减小安全处理器10a对基于功率分析的SCA的脆弱性)。

[0143] 图12是示出了根据示例实施例的安全处理器10b的框图。安全处理器10b可以类似于参考图2描述的安全处理器10,但是可以针对每个运算生成多个随机数和多个随机操作数。省略了先前参考图2给出的描述。

[0144] 参考图12,安全处理器10b可以包括随机数生成器100b和取模计算器200b。取模计算器200b可以包括随机操作数生成器210b和输出数据生成器220b。随机数生成器100b可以向随机操作数生成器210b输出第一随机数RN1。随机操作数生成器210b可以使用第一随机数RN1生成第一随机操作数ROa,输出数据生成器220b可以在对至少一些第二输入数据ID2的运算中使用第一随机操作数ROa。

[0145] 随机数生成器100b可以在对第二输入数据ID2进行运算时向随机操作数生成器210b输出与第一随机数RN1不同的第二随机数RN2。例如,随机数生成器100b可以在对第二输入数据ID2进行运算时将输出数据信号从第一随机数RN1更新为第二随机数RN2。随机操作数生成器210b可以使用第二随机数RN2生成第二随机操作数ROb,输出数据生成器220b可以在对至少一些第二输出数据ID2的运算中使用第二随机操作数ROb。

[0146] 根据本发明构思的实施例,通过在对输入数据的运算期间更新从随机数生成器100b输出的随机数,可以进一步提高第一随机操作数ROa和第二随机操作数ROb的随机性,并且可以有效地保护输入数据ID1免受SCA。

[0147] 图13是示出了根据示例实施例的安全处理器10b的操作的示图。

[0148] 参考图12和图13,第二输入数据ID2可以包括第一子输入数据SID1和第二子输入数据SID2。随机数生成器100b可以生成第一随机数RN1,并且随机操作数生成器210b可以使用第一随机数RN1计算第一随机操作数ROa。输出数据生成器220b可以在对第二输入数据ID2中的第一子输入数据SID1的运算中使用基于第一随机数RN1生成的第一随机操作数ROa。例如,输出数据生成器220b可以通过将第一子输入数据SID1与第一随机操作数ROa相乘来生成第一子输出数据SOD1。

[0149] 接下来,随机数生成器100b可以生成第二随机数RN2,并且随机操作数生成器210b可以使用第二随机数RN2计算第二随机操作数ROb。输出数据生成器220b可以在对第二输入数据ID2中的第二子输入数据SID2的运算中使用基于第二随机数RN2生成的第二随机操作

数ROb。在一个示例中，输出数据生成器220b可以通过将第二子输入数据SID2与第二随机操作数Rob相乘来生成第二子输出数据SOD2。

[0150] 输出数据生成器220b可以生成余数值作为输出数据OD，该余数值是通过将所生成的第一子输出数据SOD1和第二子输出数据SOD2按数位数相加并且通过将相加的结果除以模M而获得的。在一个示例中，输出数据生成器220b可以将第一子输出数据SOD1与通过将第二子输出数据SOD2的最低有效位和第二子输入数据SID2的最低有效位进行匹配而获得的结果相加。

[0151] 图14是示出了根据示例实施例的安全处理器10c的视图。除了安全处理器10c可以包括随机操作数生成器内的随机数寄存器以外，安全处理器10c可以类似于参考图2描述的安全处理器10。省略了先前参考图2给出的描述。

[0152] 参考图14，安全处理器10c可以包括随机数生成器100c和取模计算器200c。取模计算器200c可以包括随机操作数生成器210c和输出数据生成器220c。随机操作数生成器220c可以包括用于存储随机数的随机数寄存器223c。

[0153] 随机操作数生成器210c可以向随机数生成器100c输出随机数请求信号RSig_RN。随机数生成器100c可以将响应于随机数请求信号RSig_RN而被更新的更新后的随机数RN_upt存储在随机数寄存器223c中。

[0154] 在一个实施例中，当发生特定事件时，随机操作数生成器210c可以向随机数生成器100c输出随机数请求信号RSig_RN。在一个实施例中，特定事件可以是完成了对输入数据的重编码。在另一个实施例中，特定事件可以是已经完成了对特定数目的比特的重编码。

[0155] 在另一实施例中，随机操作数生成器210c可以在特定事件的时间点已经过去时向随机数生成器100c输出随机数请求信号RSig_RN。

[0156] 图15是示出了根据示例实施例的应用处理器1000的框图。

[0157] 参考图15，应用处理器1000可以被实现为单片系统(SoC)。应用处理器1000可以包括中央处理单元(CPU)1010、安全处理器1020、调制解调器1030、显示控制器1040、只读存储器(ROM)1050、存储控制器1060和RAM 1070。除了所示组件之外，应用处理器1000还可以包括诸如电源管理单元、图形处理单元(GPU)和时钟单元的其他组件。

[0158] CPU 1010可以处理或执行存储在ROM 1050和/或RAM 1070中的程序或数据。ROM 1050可以存储程序和/或数据。另外，RAM 1070可以临时存储程序、数据和指令。存储控制器1060可以执行与外部存储设备的接口连接，并且可以通过根据数据访问请求控制外部存储设备来读取或写入数据。另外，显示控制器1040可以通过驱动显示设备来控制屏幕的显示操作。

[0159] 根据本发明构思的实施例，安全处理器1020可以执行如本文所述的安全性操作。在一个实施例中，安全处理器1020可以包括取模计算器1021，并且可以使用随机数执行取余运算。尽管未在图15中示出，但是安全处理器1020还可以包括随机数生成器等。

[0160] 当在应用处理器1000中提供调制解调器1030时，应用处理器1000可以被称为调制解调器应用处理器(ModAP)。可以通过调制解调器1030向外部系统发送/从外部系统接收需要进行安全性计算的信息。此时，安全处理器1020可以执行根据上述实施例的安全性计算。

[0161] 如上所述，已经在附图和说明书中公开了实施例。虽然本文已经引用特定术语描述了实施例，但是应当理解的是，这些术语仅用于描述本发明构思的技术思想，而不限制如

权利要求限定的本发明构思的范围。因此，本领域普通技术人员将理解的是，在不脱离本发明构思的范围的情况下，各种修改和等同实施例都是可能的。因此，本发明构思的真正的保护范围应由所附权利要求的技术构思确定。

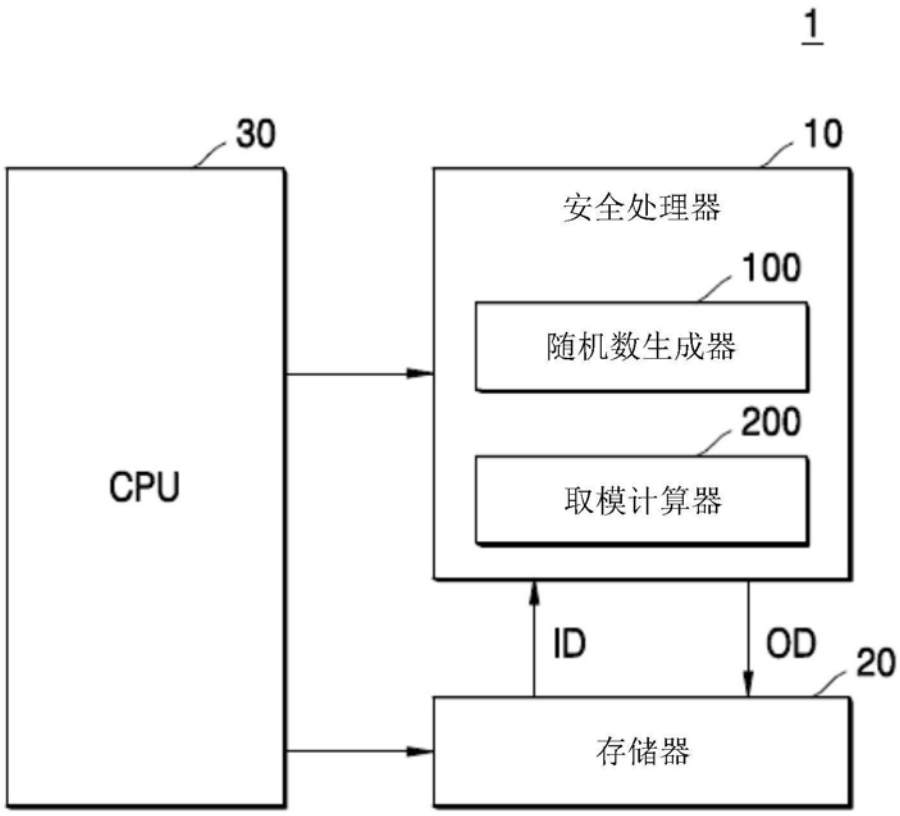

1
30
10
安全处理器
100
随机数生成器
200
取模计算器
CPU
ID
OD
20
存储器

图1

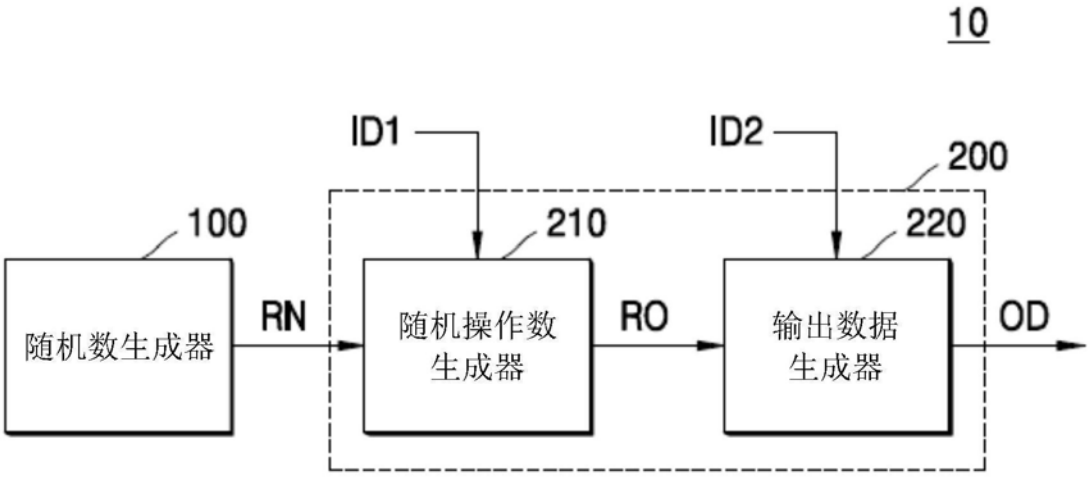

10
ID1
ID2
200
100
210
220
随机数生成器
RN
随机操作数
生成器
RO
输出数据
生成器
OD

图2

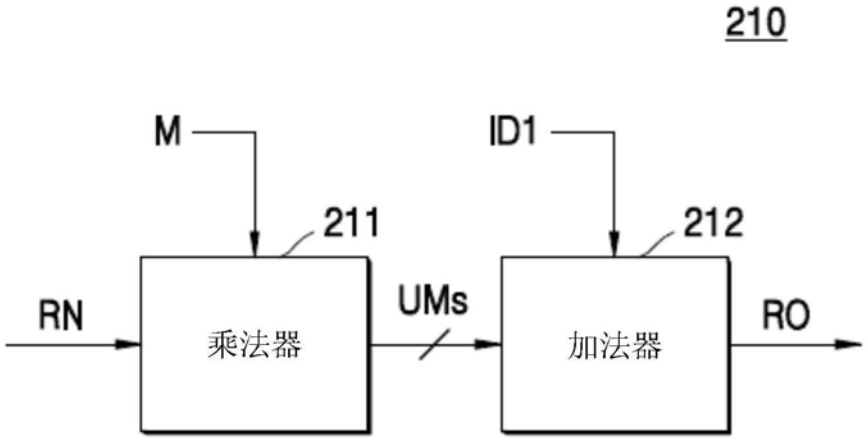

210
M
ID1
211
212
RN
乘法器
UMs
加法器
RO

图3

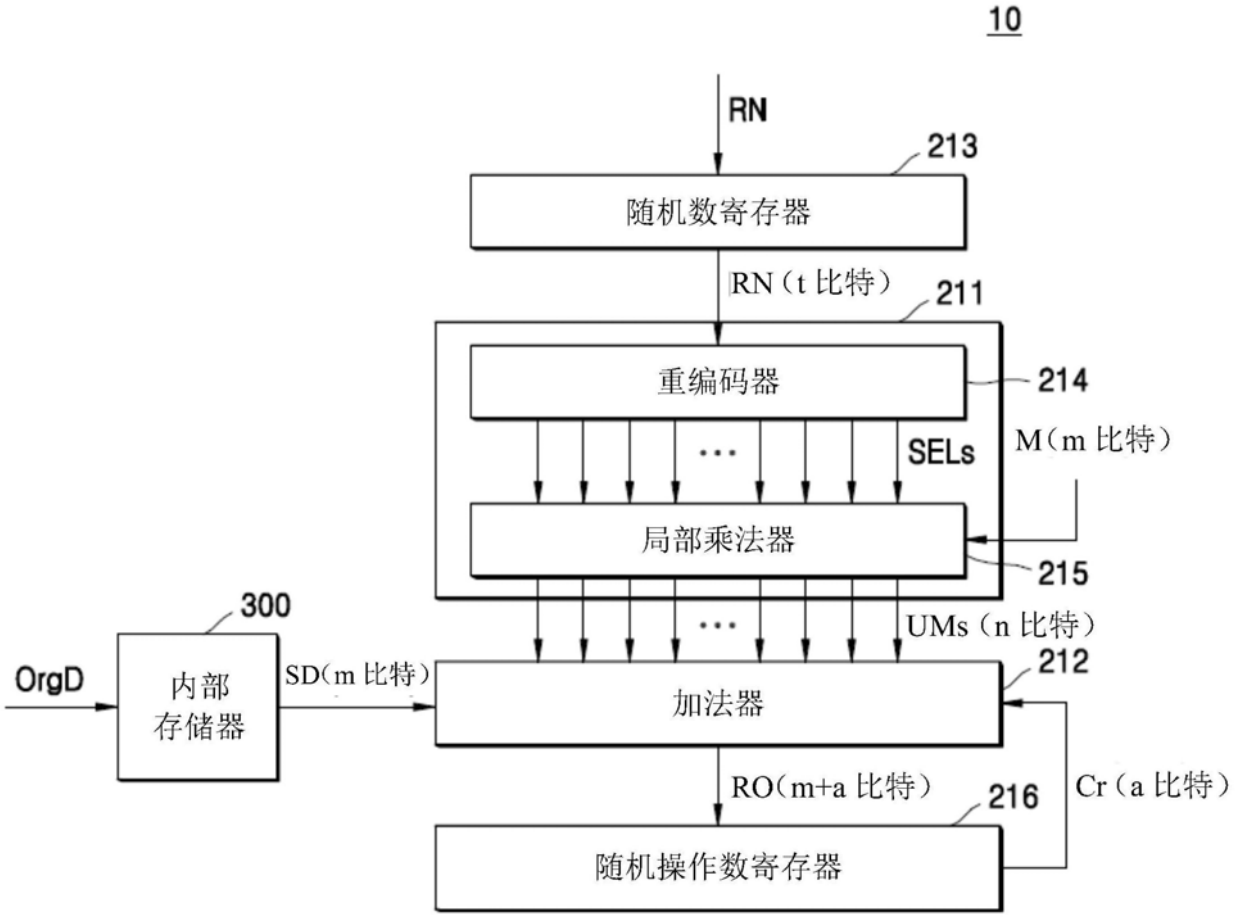

10
RN
213
随机数寄存器
RN（t 比特）
211
重编码器
214
SELs
M（m 比特）
局部乘法器
215
300
UMs（n 比特）
OrgD
内部
存储器
SD(m 比特)
212
加法器
RO(m+a 比特)
216
Cr（a 比特）
随机操作数寄存器

图4

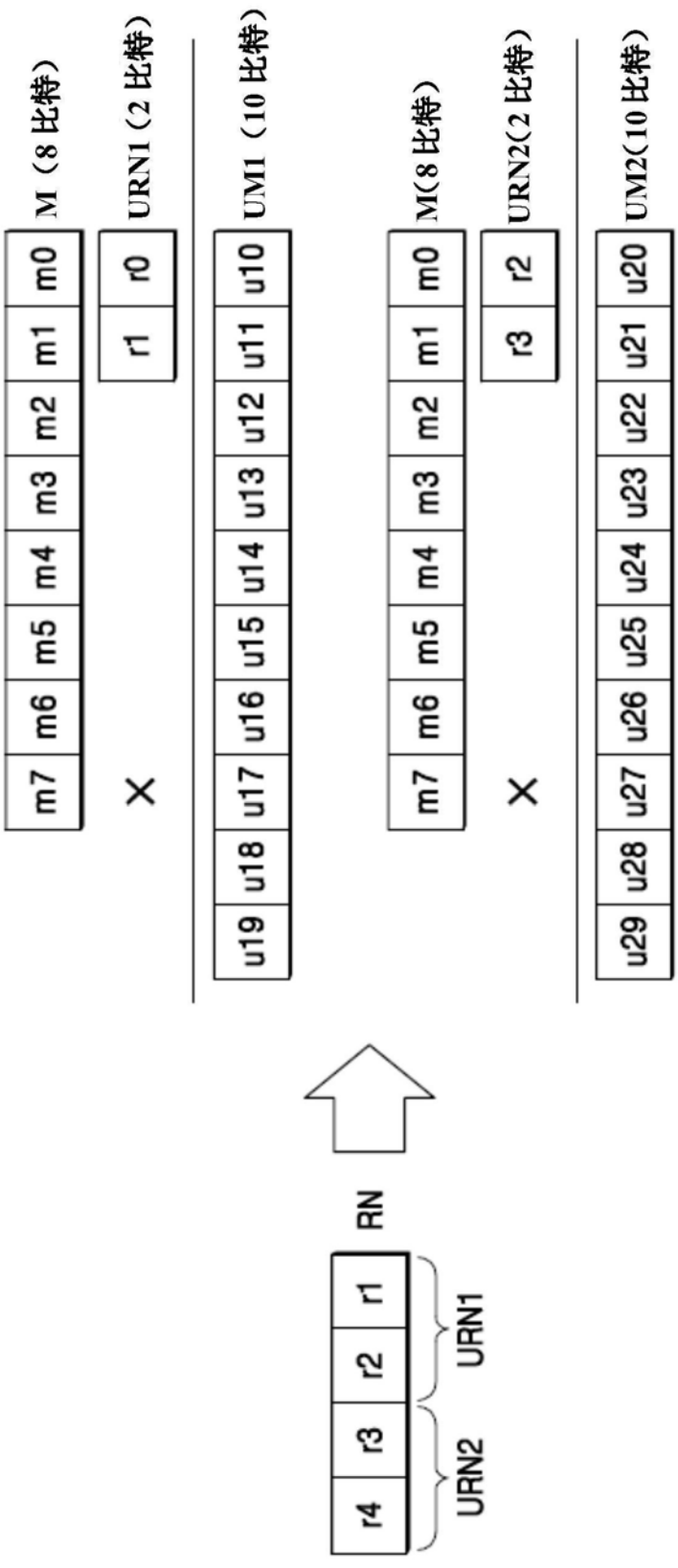

M（8 比特）
m7 m6 m5 m4 m3 m2 m1 m0
URN1（2 比特）
r1 r0
UM1（10 比特）
u19 u18 u17 u16 u15 u14 u13 u12 u11 u10
M(8 比特)
m7 m6 m5 m4 m3 m2 m1 m0
URN2(2 比特)
r3 r2
UM2(10 比特)
u29 u28 u27 u26 u25 u24 u23 u22 u21 u20
RN
r4 r3 r2 r1
URN2
URN1

图5

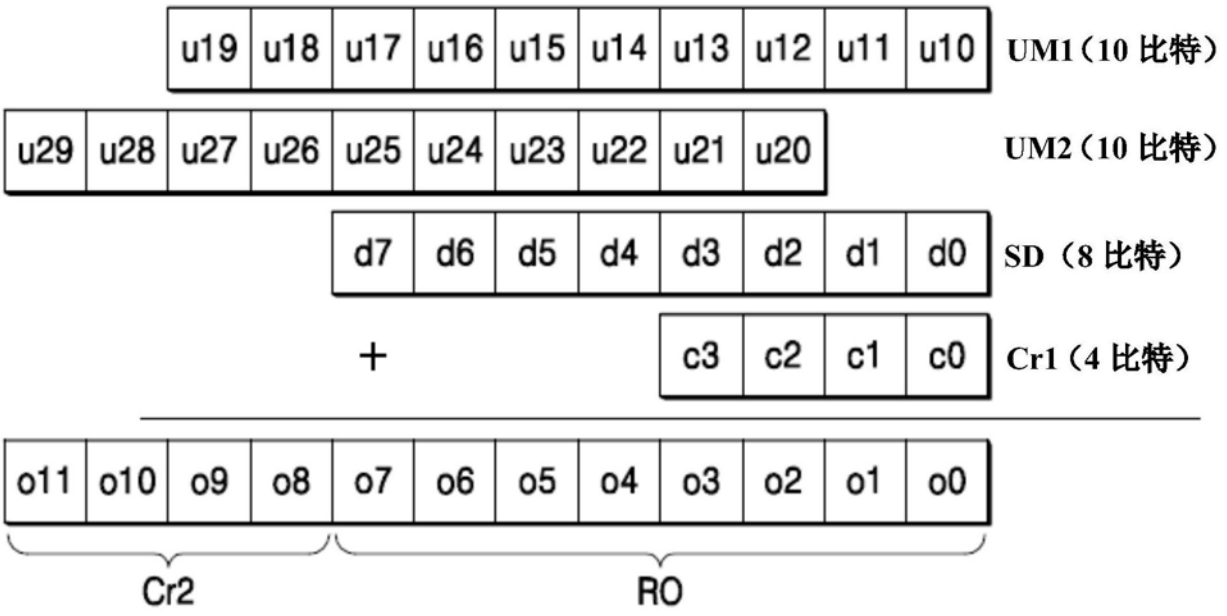

u19 u18 u17 u16 u15 u14 u13 u12 u11 u10
UM1（10 比特）
u29 u28 u27 u26 u25 u24 u23 u22 u21 u20
UM2（10 比特）
d7 d6 d5 d4 d3 d2 d1 d0
SD（8 比特）
+
c3 c2 c1 c0
Cr1（4 比特）
o11 o10 o9 o8 o7 o6 o5 o4 o3 o2 o1 o0
Cr2
RO

图6

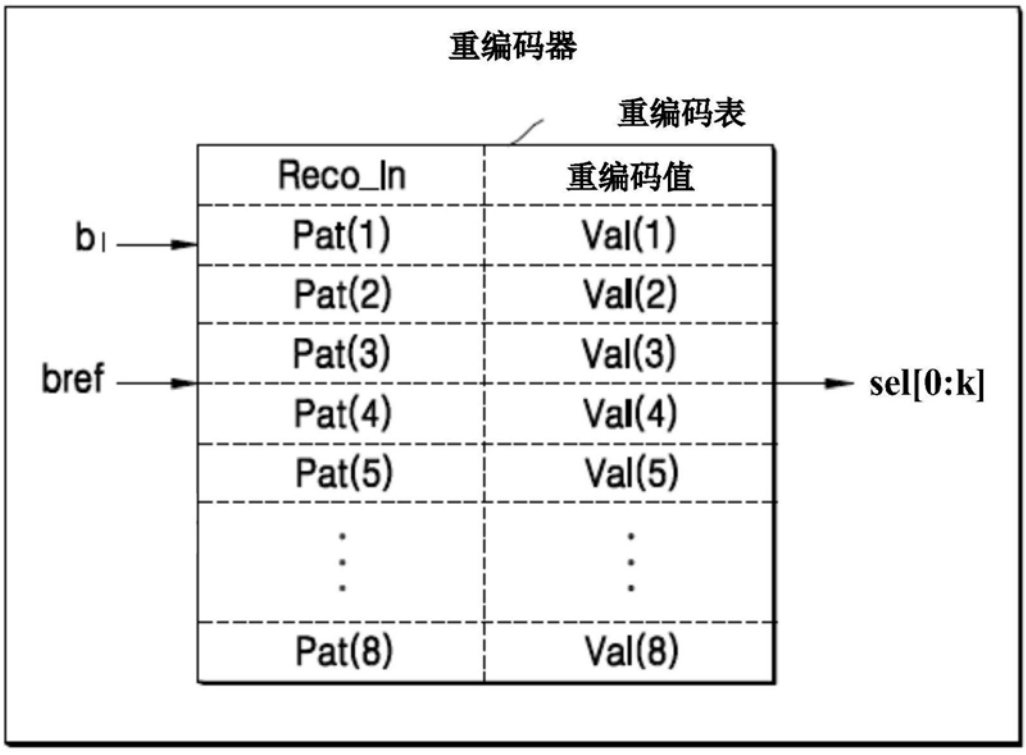

重编码器
重编码表
Reco_In
重编码值
b
Pat(1)
Val(1)
Pat(2)
Val(2)
Pat(3)
Val(3)
bref
Pat(4)
Val(4)
sel[0:k]
Pat(5)
Val(5)
Pat(8)
Val(8)

图7

b_{2i+1}	b_{2i}	b_{2i-1} (Ref bit)	Recoding value	b'_{2i+1} (Next Ref bit)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	2	0
1	0	0	−2	1
1	0	1	−1	1
1	1	0	−1	1
1	1	1	0	1

图8

Recoding Value	sel0	sel1	sel2	sel3
−2	1	0	1	0
−1	1	0	0	1
0	0	0	0	0
1	0	1	0	1
2	0	1	1	0

图9

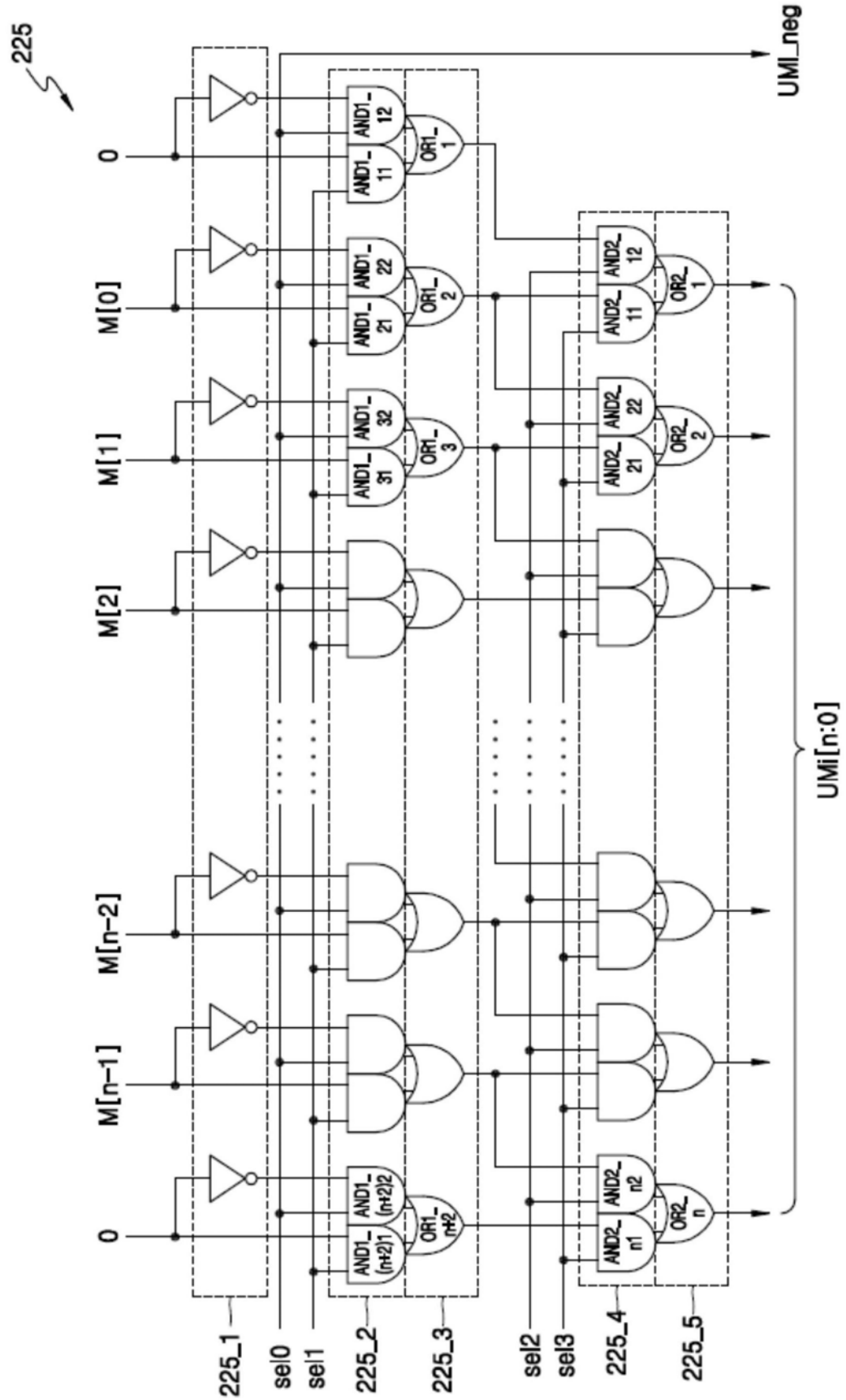

225
UMI_neg
0
M[0]
M[1]
M[2]
M[n-2]
M[n-1]
0
AND1_12
AND1_11
OR1_1
AND1_22
AND1_21
OR1_2
AND1_32
AND1_31
OR1_3
AND1_(n+2)2
AND1_(n+2)1
OR1_n+2
AND2_12
AND2_11
OR2_1
AND2_22
AND2_21
OR2_2
AND2_n2
AND2_n1
OR2_n
UMi[n:0]
225_1
sel0
sel1
225_2
225_3
sel2
sel3
225_4
225_5

图10

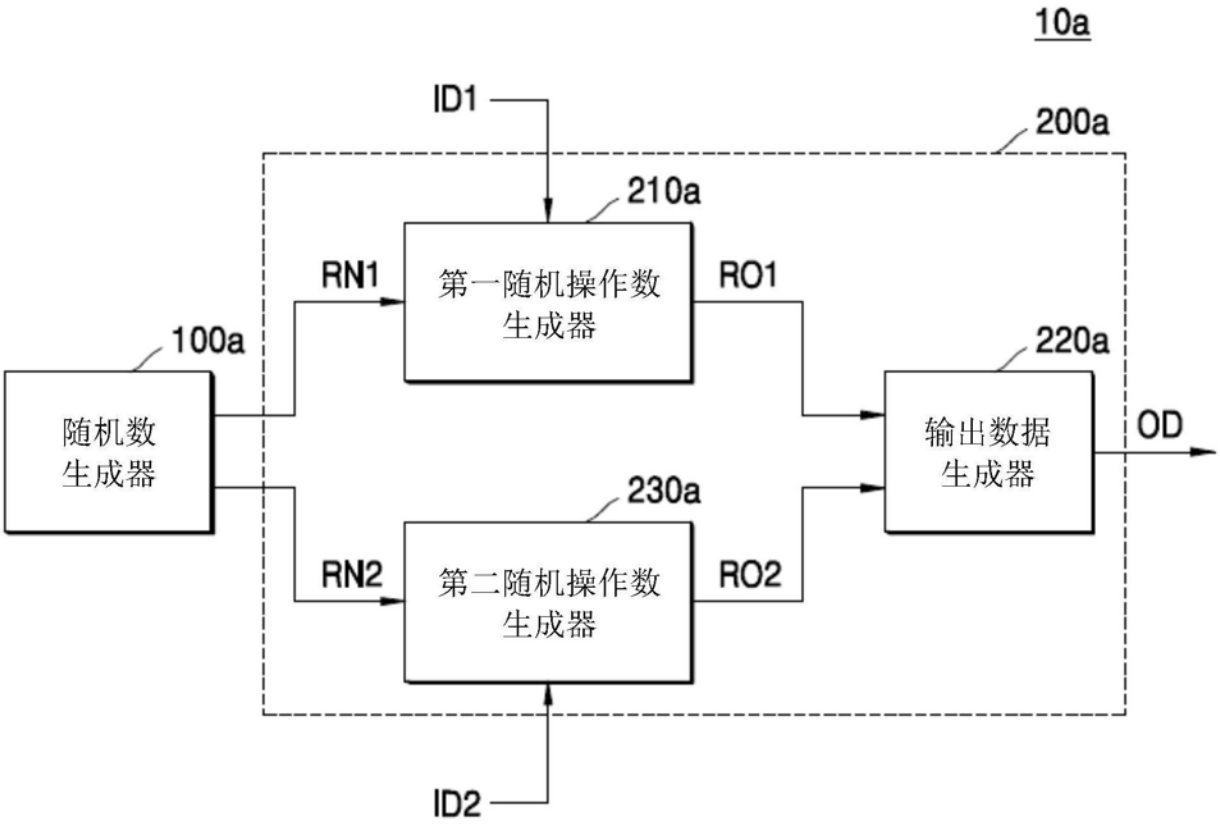

10a
ID1
200a
210a
RN1
第一随机操作数
生成器
RO1
100a
220a
随机数
生成器
输出数据
生成器
OD
230a
RN2
第二随机操作数
生成器
RO2
ID2

图11

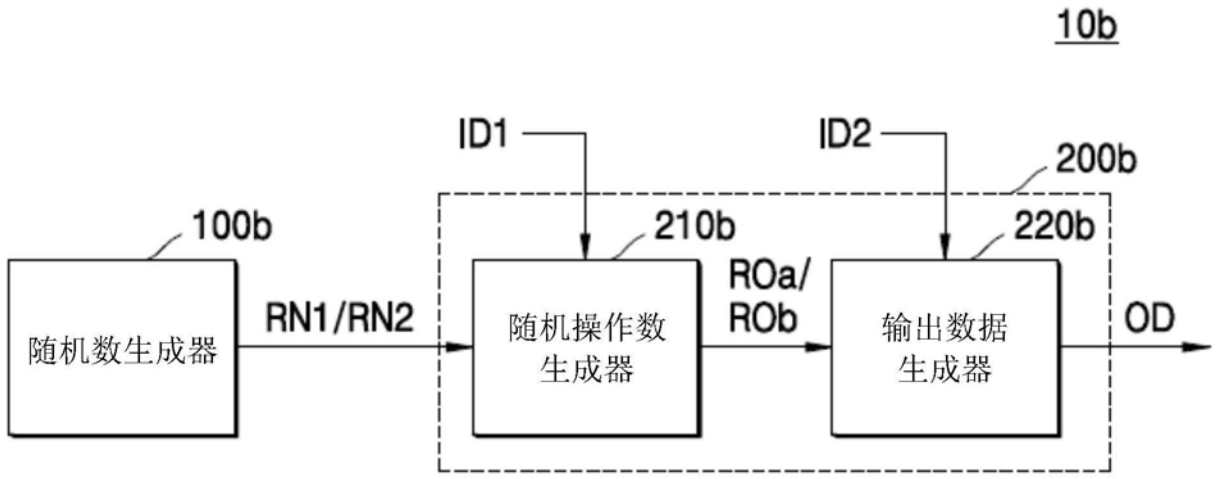

10b
ID1
ID2
200b
100b
210b
220b
ROa/
ROb
随机数生成器
RN1/RN2
随机操作数
生成器
输出数据
生成器
OD

图12

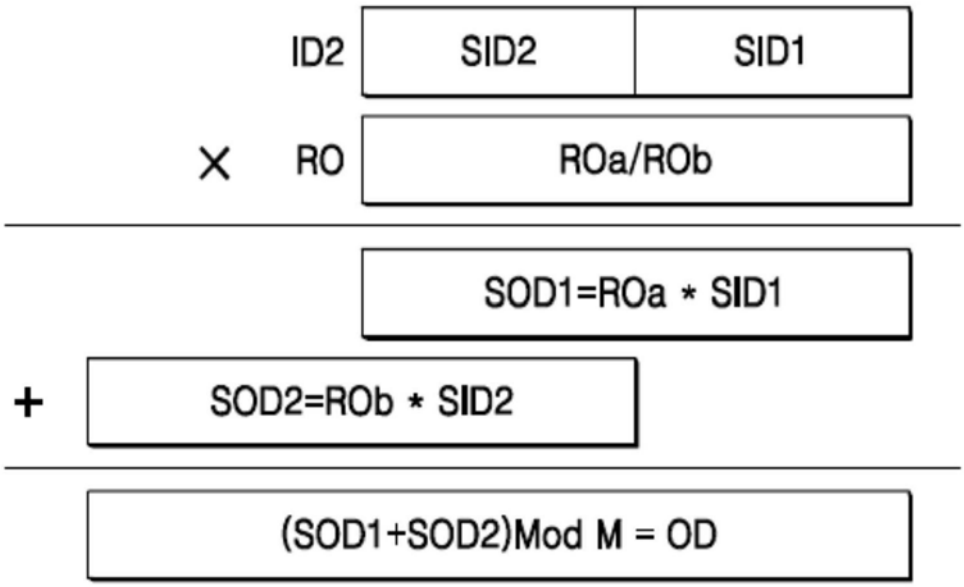

ID2
SID2
SID1
×
RO
ROa/ROb
SOD1=ROa * SID1
+
SOD2=ROb * SID2
(SOD1+SOD2)Mod M = OD

图13

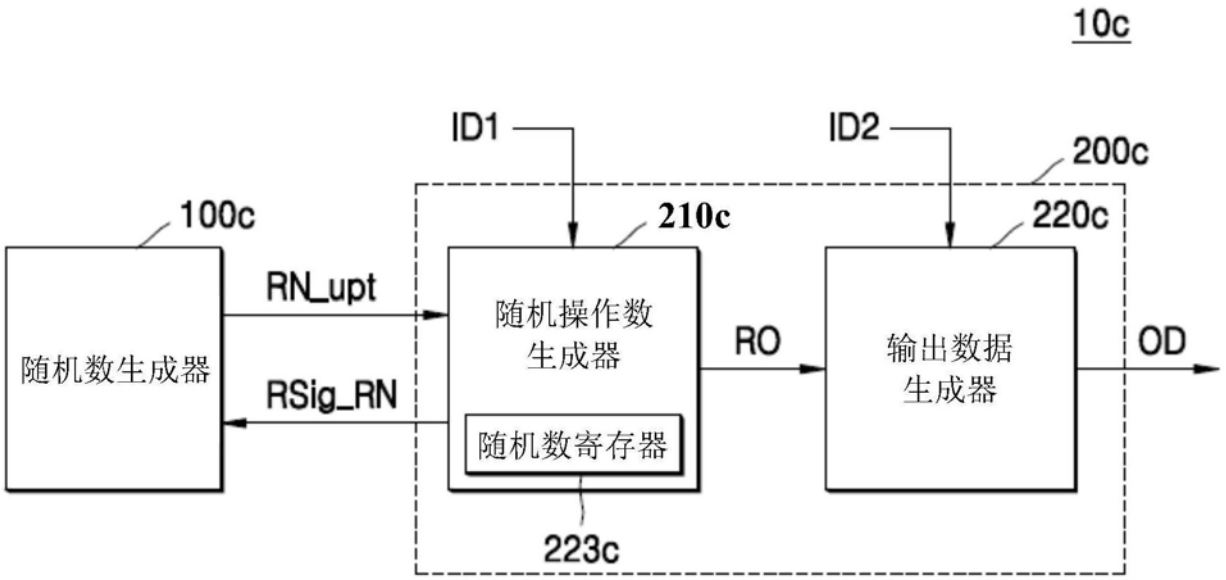

10c
ID1
ID2
200c
100c
210c
220c
随机数生成器
RN_upt
RSig_RN
随机操作数
生成器
随机数寄存器
223c
RO
输出数据
生成器
OD

图14

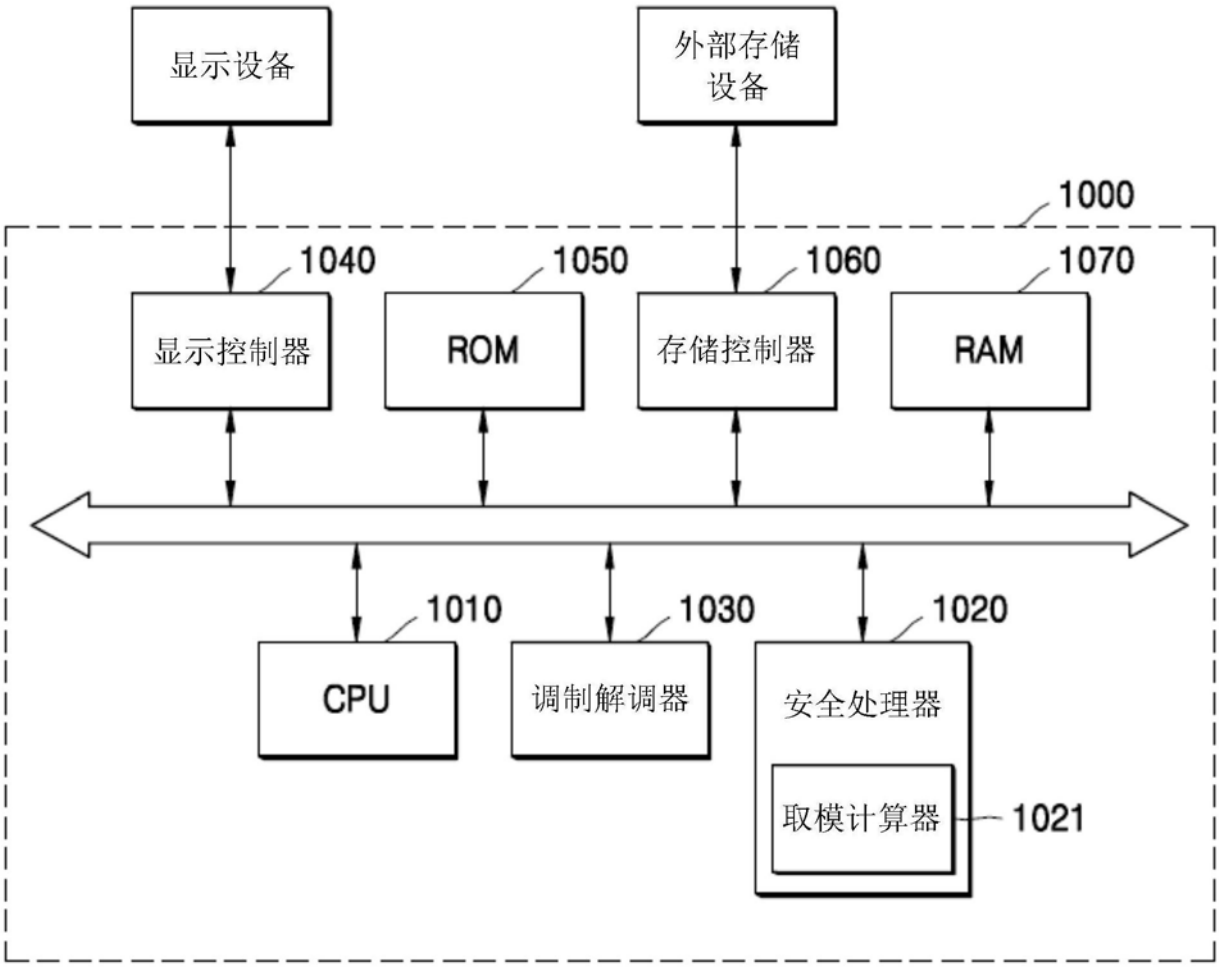

显示设备
外部存储设备
1000
1040
1050
1060
1070
显示控制器
ROM
存储控制器
RAM
1010
1030
1020
CPU
调制解调器
安全处理器
取模计算器
1021

图15