(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 109768854 B
(45)授权公告日 2020.02.04

(21)申请号 201910250703.2

(22)申请日 2019.03.29

(65)同一申请的已公布的文献号
申请公布号 CN 109768854 A

(43)申请公布日 2019.05.17

(73)专利权人 衡阳师范学院
地址 421002 湖南省衡阳市珠晖区衡花路16号衡阳师院

(72)发明人 李浪 曹夏薇

(74)专利代理机构 长沙市融智专利事务所(普通合伙) 43114
代理人 龚燕妮

(51)Int.Cl.
*H04L 9/06*(2006.01)
*H04L 9/00*(2006.01)

(56)对比文件
CN 107707343 A,2018.02.16,
CN 108123791 A,2018.06.05,
US 2012219150 A1,2012.08.30,
李浪,刘波涛.Surge:一种新型、低资源、高效的轻量级分组密码算法.《计算机科学》.2018,

审查员 马晔

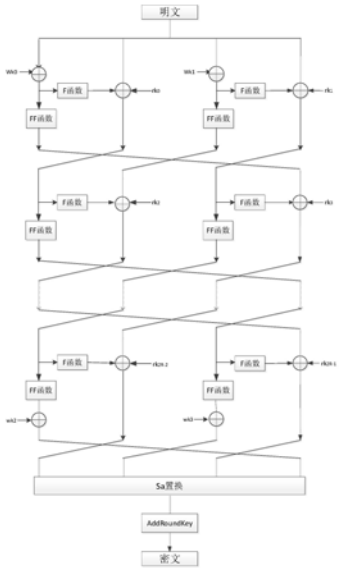
权利要求书3页 说明书14页 附图5页

(54)发明名称

一种轻量级分组密码算法的实现方法

(57)摘要

本发明公开了一种轻量级分组密码算法的实现方法,包括利用初始加密密钥计算轮数,再利用初始加密密钥依次得到中间密钥、白化密钥;并对明文进行分组,再对分组的明文进行R轮轮运算,轮运算由F函数操作、FF函数操作和按位异或操作组成,每一轮轮运算均是对N组加密信息进行处理,每下一轮轮操作对应的N组加密信息为前一轮轮操作的输出信息;最后对轮运算后的输出数据进行Sa置换操作得到输出数据,并将输出数据与初始加密密钥的低位进行轮加密操作得到加密后的明文信息。本发明通过该方法提高了加密的安全性以及效率。




CN 109768854 B

1.一种轻量级分组密码算法的实现方法,其特征在于:包括如下步骤:

S1:获取待加密的明文和初始加密密钥,并基于所述初始加密密钥计算出轮数R;

S2:按照从高位至低位顺序并依据预设分组长度对所述初始加密密钥进行分组得到中间密钥,并利用中间密钥生成N个白化密钥;

S3:按照从高位至低位顺序并依据预设分组长度对所述明文进行分组得到N组明文信息,并利用部分白化密钥对所述N组中的Ni标号组明文信息进行异或运算得到N组初始加密信息,其中,所述N组初始加密信息包括异或处理后的Ni标号组信息以及未异或处理的明文信息;

S4:利用所述N组初始加密信息进行R轮轮运算得到N组输出数据,所述轮运算由F函数操作、FF函数操作和按位异或操作组成;

其中,每一轮轮运算均是对N组加密信息进行处理,每下一轮轮操作对应的N组加密信息为前一轮轮操作的输出信息构成;所述F函数操作由轮密钥加操作、S盒替换、位混淆、异或操作组成;所述FF函数为二类广义Feistel网络结构,包括S盒和循环左移操作;

S5:基于步骤S4中N组输出数据利用步骤S3中剩余白化密钥对步骤S4中N组输出数据中Ni标号组对应数据进行异或处理得到N个数据块,所述N个数据块包括异或处理后Ni标号组对应的输出数据以及步骤S4中到N组输出数据中未标号组的数据;

S6:将步骤S5中N个数据块作为Sa置换操作的初始值进行Sa置换操作得到输出数据,并将输出数据与初始加密密钥的低位进行轮加密操作;其中,初始加密密钥的低位位数长度与输出数据的位数长度相同;

所述Sa置换是以1位为单位在16位数据块内进行置换后又以8位为单位在16位数据块之间进行置换排列的线性操作。

2.根据权利要求1所述的方法,其特征在于:步骤S4中每一轮轮运算的过程如下:

对所述N组加密信息中,利用F函数对所述N组加密信息中Ni标号组加密信息进行处理得到Ni标号组F函数输出信息,然后,将所述Ni标号组F函数输出信息分别与所述N组加密信息中非Ni标号组加密信息、当前轮运算对应的外部轮密钥进行异或处理得到非Ni标号组信息;

其中,每一轮轮运算的外部轮密钥$rk_i$是基于当前轮的轮数并利用中间密钥生成的,每一轮轮运算中一个Ni标号组F函数输出信息对应一个外部轮密钥;

对所述N组加密信息中,利用FF函数对所述N组加密信息中Ni标号组加密信息进行处理得到Ni标号组信息;

其中,所述非Ni标号组信息作为下一轮轮运算中Ni标号组信息,将Ni标号组信息作为下一轮轮运算中非Ni标号组信息。

3.根据权利要求2所述的方法,其特征在于:所述FF函数的轮数按照如下规则确定:

若轮数R满足:$15 \leqslant R < 20$,所述FF函数的轮数为8轮;

若轮数R满足:$20 \leqslant R < 25$,所述FF函数的轮数为6轮;

若轮数R满足:$25 \leqslant R \leqslant 30$,所述FF函数的轮数为4轮。

4.根据权利要求2所述的方法,其特征在于:每一次轮运算中,利用F函数对所述N组加密信息中Ni标号组加密信息进行处理得到Ni标号组F函数输出信息的执行过程如下:

S4.1:对所述N组加密信息中Ni标号组加密信息进行轮密钥加操作;

其中，所述轮密钥加操作为利用轮密钥加操作的轮密钥对Ni标号组加密信息进行异或处理；

S4.2：将步骤S4.1中轮密钥加操作的输出数据作为S盒替换的输入数据进行S盒替换运算；

S4.3：将步骤S4.2中S盒替换操作的输出数据作为位混淆的输入数据进行位混淆操作；

S4.4：将位混淆操作的输出数据作为异或操作的输入数据进行操作，输出数据为所述Ni标号组F函数输出信息。

5.根据权利要求4所述的方法，其特征在于：步骤S4.2中当前轮运算的轮数为奇数轮时，采用PRESENT算法的S盒；当前轮运算的轮数为偶数轮时，采用Piccolo算法的S盒。

6.根据权利要求4所述的方法，其特征在于：步骤S3中所述预设分组长度为16位，步骤S4.4的执行过程如下：

首先，将位混淆操作输出数据中每个标号组对应的输出数据分别从高位到低位按照4位一组划分，记为：$xor_0$、$xor_1$、$xor_2$及$xor_3$；

然后，将$xor_0$、$xor_3$分别对应与$xor_2$、$xor_1$进行异或操作得到$xor_1$'和$xor_2$'；

$$xor_1'=xor_0 \oplus xor_2,\quad xor_2'=xor_3 \oplus xor_1;$$

其次，将得到的$xor_1$'、$xor_2$'分别对应与$xor_0$、$xor_3$进行异或操作得到$xor_0$'和$xor_3$'；

$$xor_0'=xor_1' \oplus xor_0,\quad xor_3'=xor_2' \oplus xor_3;$$

最后，将得到的结果按从左至右从高位到低位的顺序排列为$xor_0$'、$xor_1$'、$xor_2$'和$xor_3$'；并依此连接输出。

7.根据权利要求4所述的方法，其特征在于：步骤S4.1中所述轮密钥加操作的轮密钥是由步骤S2中的中间密钥中任意两组中间密钥进行按位异或计算得到，其轮密钥加操作的轮密钥表示为：k[j]，j为非负整数；

步骤S4.1中选择的轮密钥加操作的轮密钥是根据轮数R选取的，规则如下：

当密钥长度为96位时，步骤S4.1中选择的轮密钥加操作的轮密钥k[j]中满足：j＝R％15；

当密钥长度为128位时，步骤S4.1中选择的轮密钥加操作的轮密钥k[j]中满足：j＝R％28。

8.根据权利要求1所述的方法，其特征在于：步骤S3中所述预设分组长度为16位，明文长度为64位，得到的所述N个数据块为4个数据块，步骤S6中将步骤S5中N个数据块作为Sa置换操作的初始值进行Sa置换操作得到输出数据的执行过程如下：

S6.1：将4个数据块从左至右标记为$P_1$、$P_2$、$P_3$和$P_4$，并分别对所述4个数据块从高位到低位依次划分为8个字节$g_0$，$g_1$，$g_2$，$g_3$，$g_4$，$g_5$，$g_6$，$g_7$；

每个数据块从高位到低位依次划分为2个字节，每个数据块均为4×4矩阵；

S6.2：分别将4个数据块$P_1$、$P_2$、$P_3$和$P_4$依数据块矩阵的两条中心对称轴划分成4个2×2的数据块矩阵，并从左上角的2×2数据块矩阵开始按照顺时针顺序分别对应给所述4个2×2的数据块矩阵编号为$A_i$、$B_i$、$D_i$和$C_i$，1≤i≤4，以及对每个2×2的数据块矩阵$A_i$、$B_i$、$D_i$和$C_i$从左上角开始按照顺时针对应编号为$a_1$、$a_2$、$a_4$和$a_3$，$b_1$、$b_2$、$b_4$和$b_3$，$d_1$、$d_2$、$d_4$和$d_3$以及$c_1$、$c_2$、$c_4$和$c_3$；

S6.3:对每个数据块中的4个2×2数据块矩阵$A_i$、$B_i$、$D_i$和$C_i$分别按照预设顺序进行置换操作,过程如下:

对于数据块$P_1$,将数据块矩阵$P_1$中$A_1$区域从$a_2$开始按逆时针连接到$a_4$结束组成$g_0$'的前半字节,接着$C_1$区域也从$c_2$开始按逆时针连接到$c_4$结束组成$g_0$'的后半字节,$B_1$区域从$b_1$开始按顺时针连接到$b_3$结束组成$g_1$'的前半字节,$D_1$区域也从$d_1$开始按顺时针连接到$d_3$结束组成$g_1$'的后半字节;

对于数据块$P_2$,将数据块矩阵$P_2$中$B_2$区域从$b_4$开始按逆时针连接到$b_3$结束组成$g_2$'的前半字节,接着$A_2$区域也从$a_4$开始按逆时针连接到$a_3$结束组成$g_2$'的后半字节,$C_2$区域从$c_1$开始按逆时针连接到$c_2$结束组成$g_3$'的前半字节,$D_2$区域也从$d_1$开始按逆针连接到$d_2$结束组成$g_3$'的后半字节;

对于数据块$P_3$,将数据块矩阵$P_3$中$C_3$区域从$c_3$开始按逆时针连接到$c_1$结束组成$g_4$'的前半字节,接着$A_3$区域也从$a_3$开始按逆时针连接到$a_1$结束组成$g_4$'的后半字节,$D_3$区域从$d_4$开始按顺时针连接到$d_2$结束组成$g_5$'的前半字节,$B_3$区域也从$b_4$开始按逆顺针连接到$b_2$结束组成$g_5$'的后半字节;

对于数据块$P_4$,将数据块矩阵$P_4$中$A_4$区域从$a_1$开始按逆时针连接到$a_2$结束组成$g_6$'的前半字节,接着$B_4$区域也从$b_1$开始按逆时针连接到$b_2$结束组成$g_6$'的后半字节,$D_4$区域从$d_4$开始按逆时针连接到$d_3$结束组成$g_7$'的前半字节,$C_4$区域也从$c_4$开始按逆顺针连接到$c_3$结束组成$g_7$'的后半字节;

S6.4:按照$g_3$',$g_7$',$g_5$',$g_6$',$g_1$',$g_4$',$g_0$',$g_2$'的顺序进行连接并输出数据形成64位数据输出。

9.根据权利要求1所述的方法,其特征在于:步骤S1中,轮数R的获取方式如下:

S1.1:提取所述初始加密密钥中高八位的值cnt;

S1.2:将步骤S1.1中值cnt对15进行取余操作;

S1.3:将步骤S1.2获得的值与基础轮数相加得到算法的轮数R,所述基础轮数为15,其中,轮数R的计算公式如下:

R=cnt%15+15。

10.根据权利要求1所述的方法,其特征在于:所述白化密钥计算公式如下:

公式a为密钥长度为96位时的白化密钥计算公式,公式b为密钥长度为128位时的白化密钥计算公式;

公式a:

$wk_0 \leftarrow k\_i[0]^L | k\_i[1]^R, wk_1 \leftarrow k\_i[1]^L | k\_i[0]^R, wk_2 \leftarrow k\_i[4]^L | k\_i[3]^R,$

$wk_3 \leftarrow k\_i[3]^L | k\_i[4]^R$

公式b:

$wk_0 \leftarrow k\_i[0]^L | k\_i[1]^R, wk_1 \leftarrow k\_i[1]^L | k\_i[0]^R, wk_2 \leftarrow k\_i[4]^L | k\_i[7]^R,$

$wk_3 \leftarrow k\_i[7]^L | k\_i[4]^R$

式中,$wk_0$、$wk_1$、$wk_2$、$wk_3$表示得到的四个白化密钥;|为连接符,$k\_i[0]^L$、$k\_i[1]^L$、$k\_i[3]^L$、$k\_i[4]^L$、$k_i[7]^L$分别表示中间密钥k_i[0]、k_i[1]、k_i[3]、k_i[4]、k_i[7]的高8位,$k\_i[0]^R$、$k\_i[1]^R$、$k\_i[3]^R$、$k\_i[4]^R$、$k_i[7]^R$分别表示中间密钥k_i[0]、k_i[1]、k_i[3]、k_i[4]、k_i[7]的低8位。

# 一种轻量级分组密码算法的实现方法

## 技术领域

[0001] 本发明属于计算机加密技术领域，具体涉及一种轻量级分组密码算法的实现方法。

## 背景技术

[0002] 近年来，轻量级加密算法凭借其密钥长度相对较短、密码算法结构简单、资源消耗小等特点成为物联网加密算法研究的重要方向之一。轻量级分组密码算法必须能够在硬件资源严格受限的硬件设备上快速执行且要保证相对的安全性。轻量级密码算法与传统密码算法相比，轻量级密码算法的执行效率更高、计算资源消耗更少，更适合于计算能力有限的RFID标签、微型无线传感器等设备。而当下许多轻量级密码已被设计和实施，以提供诸如无线传感器节点和RFID标签等应用的安全性，这些应用的限制因素是其占地面积，门电路等效数量(GE) 和功耗。

[0003] 数据加密技术是网络安全的基石，其具体实现是以加密算法为载体进行实现。分组密码具有速度快、易于标准化和便于软硬件实现等特点，通常是信息与网络安全中实现数据加密的核心体制之一。由于分组密码出现较早，各项技术成熟，为轻量级分组密码的研究提供了基础。

[0004] 随着物联网技术的不断发展，物联网环境下的数据安全也对物联网技术提出了新的问题和挑战，越来越多的轻量级分组密码算法被提出。2011年以来，国际学术界就陆续发表了一些有关轻量级分组密码算法的论文，如密码硬件与嵌入式系统国际会议提出的Piccolo，国际安全、隐私和应用密码学工程学会议(SPACE 2014)提出的Khudra等。

[0005] 目前轻量级分组密码算法仍存在以下几个问题：

[0006] (1)在有限的环境计算资源下，轻量级分组密码算法仍需要在安全性和效率之间进行权衡。一些轻量级密码算法为了抵抗差分功耗攻击和多种技术相结合的旁路攻击，把密码组件或者运算模块设计得较为复杂，从而使得算法占用较多的计算资源。

[0007] (2)一些轻量级密码算法多采用固定组件，算法不能对自身进行控制和调节。

[0008] (3)目前一些轻量级算法的加密模式比较固定，加密的轮数和运算变换模块是高度确定的，这种高度的确定性将会在一定程度上给算法带来安全隐患。比如，绝大多数的轻量级算法中，密钥长度固定，那么意味着加密算法的轮数也是固定不变的；还有一些轻量级算法的 S盒替换是直接用固定的S盒去参与运算变化，这样在一定程度上增加了算法被破解的可能性。

[0009] 因此，现有的轻量级分组密码算法的可靠性还需要进一步提高，需要安全性更高的轻量级分组密码算法。

## 发明内容

[0010] 本发明的目的是针对现有轻量级分组密码算法的安全性能还有待提高的问题，提供一种轻量级分组密码算法的实现方法，提高了加密过程的安全性以及提高了算法的效

率。

[0011] 一方面，本发明一种轻量级分组密码算法的实现方法，包括如下步骤：

[0012] S1：获取待加密的明文和初始加密密钥，并基于所述初始加密密钥计算出轮数R；

[0013] S2：按照从高位至低位顺序并依据预设分组长度对所述初始加密密钥进行分组得到中间密钥，并利用中间密钥生成N个白化密钥；

[0014] S3：按照从高位至低位顺序并依据预设分组长度对所述明文进行分组得到N组明文信息，并利用部分白化密钥对所述N组中的Ni标号组明文信息进行异或运算得到N组初始加密信息，其中，所述N组初始加密信息包括异或处理后的Ni标号组信息以及未异或处理的明文信息；

[0015] S4：利用所述N组初始加密信息进行R轮轮运算得到N组输出数据，所述轮运算由F函数操作、FF函数操作和按位异或操作组成；

[0016] 其中，每一轮轮运算均是对N组加密信息进行处理，每下一轮轮操作对应的N组加密信息为前一轮轮操作的输出信息构成；所述F函数操作由轮密钥加操作(AddRoundKey)、S盒替换(S_Box_Layer)、位混淆(Bit_Shuffle)、异或操作(XOR_Operation)组成；所述FF 函数为二类广义Feistel网络结构，包括S盒和循环左移操作；

[0017] S5：基于步骤S4中N组输出数据利用步骤S3中剩余白化密钥对步骤S4中N组输出数据中Ni标号组对应数据进行异或处理得到N个数据块，所述N个数据块包括异或处理后Ni标号组对应的输出数据以及步骤S4中到N组输出数据中未标号组的数据；

[0018] S6：将步骤S5中N个数据块作为Sa置换操作的初始值进行Sa置换操作得到输出数据，并将输出数据与初始加密密钥的低位进行轮加密操作；其中，初始加密密钥的低位位数长度与输出数据的位数长度相同。

[0019] 本发明的加密过程的轮数取决于初始密钥，每次加密过程中输入不同的初始密钥，轮数也会相应变化，解决了加密轮数高度确定所存在的安全性；同时，相比于固定轮数算法，其资源消耗差别不大并且具有一定的安全性的情况下，对相同的明文进行加密时所花费的时间要小于等于固定轮数的算法，从而提高了算法的效率，这是由于一般情况下设计密码算法的轮数时会考虑轮数余量，在保证算法不受到攻击的最低轮数的基础上，增加轮数一定的轮数来作为轮数的余量，而本算法设计的动态轮数就是在最低轮数的基础上进行动态变化。例如，密码参与模块运算的轮数固定时，一组数据加密的时间也固定，而当密码参与模块运算的轮数不固定时，在保证算法不被攻击的基础上，一组数据加密的时间只会小于等于固定轮数时所需时间。此外，在轮运算结束后通过Sa置换，又一次对分组进行调整，克服现有Feistel 结构算法中一轮迭代只能改变部分分组数据，扩散和混淆程度不高的问题。

[0020] 进一步优选，步骤S4中每一轮轮运算的过程如下：

[0021] 对所述N组加密信息中，利用F函数对所述N组加密信息中Ni标号组加密信息进行处理得到Ni标号组F函数输出信息，然后，将所述Ni标号组F函数输出信息分别与所述N组加密信息中非Ni标号组加密信息、当前轮运算对应的外部轮密钥进行异或处理得到非Ni标号组信息；

[0022] 其中，每一轮轮运算的外部轮密钥是基于当前轮的轮数并利用中间密钥生成的，每一轮轮运算中一个Ni标号组F函数输出信息对应一个外部轮密钥；

[0023] 对所述N组加密信息中，利用FF函数对所述N组加密信息中Ni标号组加密信息进行处理得到Ni标号组信息；

[0024] 其中，所述非Ni标号组信息作为下一轮轮运算中Ni标号组信息，将Ni标号组信息作为下一轮轮运算中非Ni标号组信息。

[0025] 进一步优选，所述FF函数的轮数按照如下规则确定：

[0026] 若轮数R满足：15≤R<20，所述FF函数的轮数为8轮；

[0027] 若轮数R满足：20≤R<25，所述FF函数的轮数为6轮；

[0028] 若轮数R满足：25≤R≤30，所述FF函数的轮数为4轮。

[0029] 从上述可知，FF函数的轮数与算法轮数相关，而算法轮数又与初始加密密钥有关。当外部轮数较大或较小时，FF函数的轮数都相应作出调整，从而构成一个负反馈机制。若当外部轮数较小时，待加密的明文混淆性和扩散性都不足时，算法容易遭到攻击，这时FF函数可以增加其轮数起到一个平衡作用，从而使得算法对自身进行控制和调节，这样能提高算法的安全性。

[0030] 进一步优选，每一次轮运算中，利用F函数对所述N组加密信息中Ni标号组加密信息进行处理得到Ni标号组F函数输出信息的执行过程如下：

[0031] S4.1：对所述N组加密信息中Ni标号组加密信息进行轮密钥加操作(AddRoundKey)；

[0032] 其中，所述轮密钥加操作(AddRoundKey)为利用轮密钥加操作的轮密钥对Ni标号组加密信息进行异或处理；

[0033] S4.2：将步骤S4.1中轮密钥加操作(AddRoundKey)的输出数据作为S盒替换 (S_Box_Layer)的输入数据进行S盒替换运算；

[0034] S4.3：将步骤S4.2中S盒替换操作(S_Box_Layer)的输出数据作为位混淆(Bit_Shuffle) 的输入数据进行位混淆操作(Bit_Shuffle)；

[0035] S4.4：将位混淆操作(Bit_Shuffle)的输出数据作为异或操作(XOR_Operation)的输入数据进行操作，输出数据为所述Ni标号组F函数输出信息。

[0036] 进一步优选，步骤S4.2中当前轮运算的轮数为奇数轮时，采用PRESENT算法的S盒；当前轮运算的轮数为偶数轮时，采用Piccolo算法的S盒。

[0037] 当轮数为奇数轮时，引用Piccolo算法S盒；当轮数为偶数轮时，引用PRESENT算法S盒，在一定程度上可以提高了算法的安全性。

[0038] 进一步优选，步骤S3中所述预设分组长度为16位，步骤S4.4的执行过程如下：

[0039] 首先，将位混淆操作(Bit_Shuffle)输出数据中每个标号组对应的输出数据分别从高位到低位按照4位一组划分，记为：$xor_0$、$xor_1$、$xor_2$及$xor_3$；

[0040] 然后，将$xor_0$、$xor_3$分别对应与$xor_2$、$xor_1$进行异或操作得到$xor_1$'和$xor_2$'；

[0041] $xor_1'=xor_0 \oplus xor_2$， $xor_2'=xor_3 \oplus xor_1$；

[0042] 其次，将得到的$xor_1$'、$xor_2$'分别对应与$xor_0$、$xor_3$进行异或操作得到$xor_0$'和$xor_3$'；

[0043] $xor_0'=xor_1' \oplus xor_0$， $xor_3'=xor_2' \oplus xor_3$；

[0044] 最后，将得到的结果按从左至右从高位到低位的顺序排列为$xor_0$'、$xor_1$'、$xor_2$'和

$xor_3$';并依此连接输出。

[0045] 进一步优选,步骤S4.1中所述轮密钥加操作(AddRoundKey)的轮密钥是由步骤S2中的中间密钥中任意两组中间密钥进行按位异或计算得到,轮密钥加操作(AddRoundKey)的轮密钥表示为:k[j],(0,1...j...);

[0046] 步骤S4.1中选择的轮密钥加操作(AddRoundKey)的轮密钥是根据轮数R选取的,规则如下:

[0047] 当密钥长度为96位时,步骤S4.1中选择的轮密钥加操作(AddRoundKey)的轮密钥k[j] 中满足:j=R%15;

[0048] 当密钥长度为128位时,步骤S4.1中选择的轮密钥加操作(AddRoundKey)的轮密钥k[j] 中满足:j=R%28。

[0049] 进一步优选,步骤S3中所述预设分组长度为16位,明文长度为64位,得到的所述N个数据块为4个数据块,步骤S6中将步骤S5中N个数据块作为Sa置换操作的初始值进行 Sa置换操作得到输出数据的执行过程如下:

[0050] S6.1:将4个数据块从左至右标记为$P_1$、$P_2$、$P_3$和$P_4$,并分别对所述4个数据块从高位到低位依次划分为8个字节$g_0$,$g_1$,$g_2$,$g_3$,$g_4$,$g_5$,$g_6$,$g_7$;

[0051] 每个数据块从高位到低位依次划分为2个字节,每个数据块均为4×4矩阵;

[0052] 数据块$P_1$从高位到低位依次划分为8个字节$g_0$,$g_1$,数据块$P_2$从高位到低位依次划分为 8个字节$g_2$,$g_3$,数据块$P_3$从高位到低位依次划分为8个字节$g_4$,$g_5$,数据块$P_4$从高位到低位依次划分为8个字节$g_6$,$g_7$;

[0053] S6.2:分别将4数据块$P_1$、$P_2$、$P_3$和$P_4$依数据块矩阵的两条中心对称轴划分成4个2×2 的数据块矩阵,并从左上角的2×2数据块矩阵开始按照顺时针顺序分别对应给所述4个2×2 的数据块矩阵编号为$A_i$、$B_i$、$D_i$和$C_i$($1 \leqslant i \leqslant 4$),以及对每个2×2的数据块矩阵$A_i$、$B_i$、$D_i$和 $C_i$从左上角开始按照顺时针对应编号为$a_1$、$a_2$、$a_4$和$a_3$,$b_1$、$b_2$、$b_4$和$b_3$,$d_1$、$d_2$、$d_4$和$d_3$以及$c_1$、$c_2$、$c_4$和$c_3$;

[0054] S6.3:对每个数据块中的4个2×2数据块矩阵$A_i$、$B_i$、$D_i$和$C_i$分别按照预设顺序进行置换操作,过程如下:

[0055] 对于数据块$P_1$,将数据块矩阵$P_1$中$A_1$区域从$a_2$开始按逆时针连接到$a_4$结束组成$g_0$'的前半字节,接着$C_1$区域也从$c_2$开始按逆时针连接到$c_4$结束组成$g_0$'的后半字节,$B_1$区域从$b_1$开始按顺时针连接到$b_3$结束组成$g_1$'的前半字节,$D_1$区域也从$d_1$开始按顺时针连接到$d_3$结束组成$g_1$'的后半字节;

[0056] 对于数据块$P_2$,将数据块矩阵$P_2$中$B_2$区域从$b_4$开始按逆时针连接到$b_3$结束组成$g_2$'的前半字节,接着$A_2$区域也从$a_4$开始按逆时针连接到$a_3$结束组成$g_2$'的后半字节,$C_2$区域从$c_1$开始按逆时针连接到$c_2$结束组成$g_3$'的前半字节,$D_2$区域也从$d_1$开始按逆针连接到$d_2$结束组成$g_3$'的后半字节;

[0057] 对于数据块$P_3$,将数据块矩阵$P_3$中$C_3$区域从$c_3$开始按逆时针连接到$c_1$结束组成$g_4$'的前半字节,接着$A_3$区域也从$a_3$开始按逆时针连接到$a_1$结束组成$g_4$'的后半字节,$D_3$区域从$d_4$开始按顺时针连接到$d_2$结束组成$g_5$'的前半字节,$B_3$区域也从$b_4$开始按逆顺针连接到$b_2$结束组成$g_5$'的后半字节;

[0058] 对于数据块$P_4$,将数据块矩阵$P_4$中$A_4$区域从$a_1$开始按逆时针连接到$a_2$结束组成$g_6$'

的前半字节，接着$B_4$区域也从$b_1$开始按逆时针连接到$b_2$结束组成$g_6$'的后半字节，$D_4$区域从$d_4$开始按逆时针连接到$d_3$结束组成$g_7$'的前半字节，$C_4$区域也从$c_4$开始按逆顺针连接到$c_3$结束组成$g_7$'的后半字节；

[0059] S6.4：按照$g_3$'，$g_7$'，$g_5$'，$g_6$'，$g_1$'，$g_4$'，$g_0$'，$g_2$'的顺序进行连接并输出数据形成 64位数据输出。

[0060] Sa置换是在待加密明文经过轮运算后又一次对其进行从字节到位的置换，从而增加扩散性，提高算法的安全性，不易遭到攻击。因此，本发明在轮运算后加入Sa置换，相较于其他基于Feistel结构的轻量级算法安全性和加密性更优越。

[0061] 进一步优选，步骤S1中，轮数R的获取方式如下：

[0062] S1.1：提取所述初始加密密钥中高八位的值cnt；

[0063] S1.2：将步骤S1.1中值cnt对15进行取余操作；

[0064] S1.3：将步骤S1.2获得的值与基础轮数相加得到算法的轮数R，所述基础轮数为15。其中，轮数R的计算公式如下：

[0065] R＝cnt％15+15。

[0066] 算法的轮数取决于初始密钥的高8位的值，即算法轮数是动态的。当对待加密的明文进行加密时，本算法的轮数是在一定区间内对待加密的明文进行加密，即算法所需时间也是在一定区间内取值，相较于固定轮数，动态轮数所需时间只会小于等于固定轮数所需时间，从而提高了算法加密的效率。

[0067] 进一步优选，所述白化密钥的计算公式如下：

[0068] 公式a为密钥长度为96位时的白化密钥计算公式，公式b为密钥长度为128位时的白化密钥计算公式；

[0069] 公式a：

[0070] $wk_0 \leftarrow k\_i[0]^L | k\_i[1]^R, wk_1 \leftarrow k\_i[1]^L | k\_i[0]^R, wk_2 \leftarrow k\_i[4]^L | k\_i[3]^R,$

[0071] $wk_3 \leftarrow k\_i[3]^L | k\_i[4]^R$

[0072] 公式b：

[0073] $wk_0 \leftarrow k\_i[0]^L | k\_i[1]^R, wk_1 \leftarrow k\_i[1]^L | k\_i[0]^R, wk_2 \leftarrow k\_i[4]^L | k\_i[7]^R,$

[0074] $wk_3 \leftarrow k\_i[7]^L | k\_i[4]^R$

[0075] 式中，$wk_0$、$wk_1$、$wk_2$、$wk_3$表示得到的四个白化密钥；|为连接符，$k\_i[0]^L$、$k\_i[1]^L$、$k\_i[3]^L$、$k\_i[4]^L$、$k_i[7]^L$分别表示中间密钥k_i[0]、k_i[1]、k_i[3]、k_i[4]、k_i[7] 的高8位，$k\_i[0]^R$、$k\_i[1]^R$、$k\_i[3]^R$、$k\_i[4]^R$、$k_i[7]^R$分别表示中间密钥k_i[0]、k_i[1]、 k_i[3]、k_i[4]、k_i[7]的低8位。

[0076] 本发明若得到N组加密信息是表示4组，则在步骤S4中涉及的外部轮密钥按照如下规则计算，本发明中每一轮轮运算需要两个外部轮密钥。

[0077] 公式c为密钥长度为96位时外部轮密钥计算公式，公式d为密钥长度为128位时外部轮密钥计算公式：

[0078] 公式c：

[0079] $$(rk_{2i},rk_{2i+1}) \leftarrow (con_{2i}^{96},con_{2i+1}^{96}) \oplus \begin{cases} (k\_i[2],k\_i[3]) & if \quad i \bmod 5 = 0 or 2 \\ (k\_i[0],k\_i[1]) & if \quad i \bmod 5 = 1 or 4 \\ (k\_i[4],k\_i[4]) & if \quad i \bmod 5 = 3 \end{cases}$$

[0080] $$(con_{2i}^{96} \mid con_{2i+1}^{96}) \leftarrow (c_{i+1} \mid c_0 \mid c_{i+1} \mid \{00\}_2 \mid c_{i+1} \mid c_0 \mid c_{i+1} \mid) \oplus \{a96cd4b8\}_{16}$$

[0081] 公式d：

[0082] $$\left.\begin{array}{l} rk_{2i} \leftarrow \text{k\_i}[(2\text{i}+2) \bmod 8] \oplus con_{2i}^{128} \\ rk_{2i+1} \leftarrow \text{k\_i}[(2\text{i}+3) \bmod 8] \oplus con_{2i+1}^{128} \end{array}\right\}$$

[0083] $$(con_{2i}^{128} \mid con_{2i+1}^{128}) \leftarrow \left(c_{i+1} \mid c_0 \mid c_{i+1} \mid \{00\}_2 \mid c_{i+1} \mid c_0 \mid c_{i+1} \mid\right) \oplus \{8b79a465\}_{16}$$

[0084] 式中，$rk_{2i}$，$rk_{2i+1}$表示当前轮数对应的外部轮密钥，|为连接符，⊕为异或符，mod代表取余运算，k_i[0]、k_i[2]、k_i[1]、k_i[3]、k_i[4]分别表示中间密钥；

[0085] $\{\}_{16}$为十六进制数，$\{\}_2$为二进制数，i为当前轮数(i的取值是0至R-1)，$c_i$是用5位二进制数表示的十进制数i。

[0086] 从上述来看，本发明针对密钥是96位或者128位提供了不同的计算公式。且由于密钥长度不同，按照预设分组长度生成的中间密钥也必然不同，譬如，预设分组长度为16时，若当密钥长度为96位时，将其从高位开始按16位一组分成6组，得到中间密钥k_i[0]、k_i[1]、 k_i[2]、k_i[3]、k_i[4]、k_i[5]。若当密钥长度为128位时，将其从高位开始按16位一组分成8组，得到中间密钥k_i[0]、k_i[1]、k_i[2]、k_i[3]、k_i[4]、k_i[5]、k_i[6]、 k_i[7]。

[0087] 有益效果

[0088] 1、本发明算法的轮数是利用初始加密密钥高八位的值，将其对15进行取余运算，取余得到的结果加上基础轮数15轮即为该次加密轮数，从而对算法轮数进行控制。一般情况下设计密码算法的轮数时会考虑轮数余量，在保证算法不受到攻击的最低轮数的基础上，增加轮数一定的轮数来作为轮数的余量，而本算法设计的动态轮数就是在最低轮数的基础上进行动态变化提高加密的效率。这是因为密码参与模块运算的轮数固定时，一组数据加密的时间也固定，而当密码参与模块运算的轮数不固定时，在保证算法不被攻击的基础上，一组数据加密的时间只会小于等于固定轮数时所需时间。

[0089] 2、本发明算法的轮数是根据初始加密密钥生成的，每次计算时输入的初始加密密钥不同，则轮数不同，且一旦变化，算法的轮数则变化。相较于一些采用固定组件的轻量级密码算法，本发明能够对自身进行控制与调节；同时FF函数轮数又取决于算法的轮数，形成了一个负反馈机制，尤其是引入本发明的FF函数，当外部轮数较小时，FF函数中的轮数就可以相对取一个较大的值，当外部轮数较大时，FF函数中的轮数就可以相对取一个较小的值，避免外部轮数较小而导致待加密的明文扩散和混淆程度不够，算法安全性不高的问题。

[0090] 3、本发明在轮运算后加入了Sa置换，在一定程度上提高了算法的扩散性，相比于其他基于Feistel结构的轻量级算法安全性和加密性更优越。同时，在F函数中引用Piccolo和 PRESENT两个算法的S盒，当轮数为奇数轮时，引用Piccolo算法S盒；当轮数为偶数轮时，引用PRESENT算法S盒，这样在一定程度上提高了算法的安全性。

[0091] 4、本发明在资源占用不大的情况下，能够抵抗差分功耗攻击，提高了算法的安全

性。这是基于差分功耗攻击是通过监测硬件装置的功耗曲线，利用统计的方法对所收集到的曲线进行分析处理，因此把密码组件设计的较为复杂而导致资源占用过多的情况，而本算法在硬件资源占用不多的情况下，提出了一种新的Sa置换方式，没有使用16位为单位的字循环移位，而是以1位为单位进行置换后又以8位为单位进行置换排列，这样拆除了16位的字结构，提高了算法的抗统计分析能力，从而能够抵抗差分功耗攻击，提高算法的安全性。

## 附图说明

[0092] 图1为本发明所述加密过程示意图；

[0093] 图2为本发明所述解密过程示意图；

[0094] 图3为本发明F函数运算过程示意图；

[0095] 图4为本发明F函数运算中Bit_Shuffle过程示意图；

[0096] 图5为本发明F函数运算中XOR_Operation过程示意图；

[0097] 图6为本发明FF函数运算过程示意图；

[0098] 图7为本发明Sa置换运算过程示意图。

## 具体实施方式

[0099] 下面将结合实施例对本发明做进一步的说明。

[0100] 本发明实施例提供的一种轻量级分组密码算法的实现方法是用于对明文进行加密，其中，本发明所述算法的命名为Wheel，本实施例中明文的长度为64位，密钥的长度分为96位和 128位两种，其分别在15轮到30轮和36轮到48轮之间进行轮函数迭代。本发明中轮函数包括F函数和FF函数，如图1所示。

[0101] F轮函数包含：轮密钥加(AddRoundkey)、S盒替换(S_Box_Layer)、位混淆(Bit_Shuffle) 和异或操作(XOR_Operation)四个模块。

[0102] FF轮函数基于广义Feistel网络结构，包含异或操作和循环左移操作。

[0103] 本发明实施例提供的一种轻量级分组密码算法的实现方法，包括如下步骤：

[0104] S1：获取待加密的明文和初始加密密钥，并基于所述初始加密密钥计算出轮数R。

[0105] 其中，本实施例中，待加密的明文为64位。并依据初始加密密钥确定轮数R，规则如下：轮数R取决于该次加密过程中输入密钥数据的高八位的值，取当前加密过程中输入密钥高八位的值cnt，再将值cnt对15进行取余，得到的结果加上基础轮数15轮为该次加密过程的轮数值。

[0106] 轮数R的计算公式如下：

[0107] R＝cnt％15+15

[0108] S2：按照从高位至低位顺序并依据预设分组长度对所述初始加密密钥进行分组得到中间密钥，并利用中间密钥生成N个白化密钥。

[0109] 本实施例中，预设分组长度为16位。当初始加密密钥长度为96位时，将其从高位开始按16位一组分成6组，得到中间密钥k_i[0]、k_i[1]、k_i[2]、k_i[3]、k_i[4]、k_i[5]。在这6组16位中间密钥中任意取出两组进行按位异或，这种情况下可以得到15组F函数中轮密钥加操作(AddRoundKey)的轮密钥，分别为k[0],k[1],...,k[13],k[14]。

[0110] 若当密钥长度为128位时，将其从高位开始按16位一组分成8组，得到中间密钥k_i[0]、k_i[1]、k_i[2]、k_i[3]、k_i[4]、k_i[5]、k_i[6]、k_i[7]。在这8组16位中间密钥中任意取出两组进行按位异或，这种情况下可以得到28组F函数中轮密钥加操作(AddRoundKey)的轮密钥，分别为k[0],k[1],...,k[26],k[27]。

[0111] 然后依据中间密钥可以生成白化密钥和外部轮密钥。其中，白化密钥的计算公式为，公式a为密钥长度为96位时的白化密钥计算公式，公式b为密钥长度为128位时的白化密钥计算公式。其中，本实施例中利用中间密钥生成4个白化密钥。

[0112] 公式a：

[0113] $wk_0 \leftarrow k\_i[0]^L | k\_i[1]^R, wk_1 \leftarrow k\_i[1]^L | k\_i[0]^R, wk_2 \leftarrow k\_i[4]^L | k\_i[3]^R,$

[0114] $wk_3 \leftarrow k\_i[3]^L | k\_i[4]^R$

[0115] 公式b：

[0116] $wk_0 \leftarrow k\_i[0]^L | k\_i[1]^R, wk_1 \leftarrow k\_i[1]^L | k\_i[0]^R, wk_2 \leftarrow k\_i[4]^L | k\_i[7]^R,$

[0117] $wk_3 \leftarrow k\_i[7]^L | k\_i[4]^R$

[0118] 式中，$wk_0$、$wk_1$、$wk_2$、$wk_3$表示得到的四个白化密钥；|为连接符，$k\_i[0]^L$、$k\_i[1]^L$、$k\_i[3]^L$、$k\_i[4]^L$、$k_i[7]^L$分别表示中间密钥k_i[0]、k_i[1]、k_i[3]、k_i[4]、k_i[7] 的高8位，$k\_i[0]^R$、$k\_i[1]^R$、$k\_i[3]^R$、$k\_i[4]^R$、$k_i[7]^R$分别表示中间密钥k_i[0]、k_i[1]、k_i[3]、k_i[4]、k_i[7]的低8位。

[0119] S3：按照从高位至低位顺序并依据16位对所述明文进行分组得到4组明文信息X[0]、X[1]、X[2]、X[3]，并利用部分白化密钥$wk_0$、$wk_1$对所述4组中的Ni标号组明文信息X[0]、X[2]进行异或运算得到$X_1[0]$和$X_1[2]$，最终得到4组初始加密信息$X_1[0]$、X[1]、$X_1[2]$及 X[3]。本实施例中，Ni标号组表示X[0]、X[2]。

[0120] 其中，具体是将X[0]与$wk_0$、X[2]与$wk_1$进行异或运算获得$X_1[0]$和$X_1[2]$。

[0121] S4：利用所述4组初始加密信息$X_1[0]$、X[1]、$X_1[2]$及X[3]进行R轮轮运算得到N组输出数据。轮运算由F函数操作、FF函数操作和按位异或操作组成。

[0122] 每一轮的运算过程包括F函数操作、FF函数操作和按位异或操作，具体过程如下：

[0123] 首先关于F函数操作部分：

[0124] S4.1：对所述4组加密信息中$X_1[0]$、$X_1[2]$组加密信息进行轮密钥加操作(AddRoundKey)；

[0125] 前述已给出，密钥为96位时，轮密钥加操作(AddRoundKey)的轮密钥为15组，分别表示为：k[0],k[1],...,k[13],k[14]；密钥为128位时，轮密钥加操作(AddRoundKey)的轮密钥为28组，分别表示为：k[0],k[1],...,k[26],k[27]。本步骤中根据轮数R选取轮密钥加操作(AddRoundKey)的轮密钥，规则如下：

[0126] 当密钥长度为96位时，步骤S4.1中选择的轮密钥加操作(AddRoundKey)的轮密钥k[j] 中满足：j＝R％15；

[0127] 当密钥长度为128位时，步骤S4.1中选择的轮密钥加操作(AddRoundKey)的轮密钥k[j] 中满足：j＝R％28。

[0128] 轮密钥加操作(AddRoundKey)是利用其轮密钥k[j]对$X_1[0]$、$X_1[2]$组加密信息进行异或处理得到$X_{add}[0]$、$X_{add}[2]$；即 $X_{add}[0] = X_1[0] \oplus k[j]$， $X_{add}[2] = X_1[2] \oplus k[j]$。

[0129] S4.2：将步骤S4.1中轮密钥加操作(AddRoundKey)输出$X_{add}[0]$、$X_{add}[2]$的作为S 盒

替换(S_Box_Layer)的输入数据进行S盒替换运算(S_Box_Layer)得到$X_{sbox}[0]$、$X_{sbox}[2]$。

[0130] S盒替换(S_Box_Layer)是一个基于S盒的非线性置换，它用于输入或中间态每一个字节通过一个简单的查表操作，将其映射为另一个字节。映射方法是把输入的数通过查表对应输出的数，而F函数中S盒替换(S_Box_Layer)运算表达式为：$X_{sbox}[0]=S(X_{add}[0])$，$X_{sbox}[2]=S(X_{add}[2])$。本实施例中，器件S盒在奇数轮是采用的PRESENT算法的S盒，记作sbox1，在偶数轮是采用的Piccolo算法的S盒，记作sbox2。该操作使得算法在奇数轮和偶数轮所使用的扩散器件不同从而在一定程度上提高了算法的安全性。本发明实施例中，F 函数中使用的S盒和FF函数中使用的S盒实现。其中，在F函数中，用PRESENT算法加密的S盒如下表1所示：

[0131] 表1 sbox1盒元素

[0132]

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S1[x] | C | 5 | 6 | B | 9 | 0 | A | D | 3 | E | F | 8 | 4 | 7 | 1 | 2 |

[0133] 在F函数中，用Piccolo算法加密的S盒如下表2所示：

[0134] 表2 sbox2盒元素

[0135]

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S2[x] | E | 4 | B | 2 | 3 | 8 | 0 | 9 | 1 | A | 7 | F | 6 | C | 5 | D |

[0136] S4.3:将步骤S4.2中S盒替换操作(S_Box_Layer)输出数据$X_{sbox}[0]$、$X_{sbox}[2]$作为位混淆(Bit_Shuffle)的输入数据进行位混淆操作(Bit_Shuffle)得到$X_{bs}[0]$、$X_{bs}[2]$。

[0137] 位混淆(Bit_Shuffle)具体步骤如下：

[0138] 步骤4.3.1:将输入的数据$X_{sbox}[0]$(或$X_{sbox}[2]$)从高位到低位按8位一组划分成2个部分，从高位到低位依次记作$S_0$和$S_1$；

[0139] 步骤4.3.2:把输入数据中的高八位移到低八位，即将$S_0$置换到初始$S_1$的位置上，$S_1$置换到初始$S_0$位置上；

[0140] 步骤4.3.3:将置换后的2组8位数据连接成16位输出得到$X_{bs}[0]$(或$X_{bs}[2]$)。

[0141] 应当理解，本发明利用步骤4.3.1-步骤4.3.3分别对输出数据$X_{sbox}[0]$、$X_{sbox}[2]$进行处理。

[0142] S4.4:将位混淆操作(Bit_Shuffle)输出数据$X_{bs}[0]$、$X_{bs}[2]$作为异或操作(XOR_Operation) 的输入数据进行操作，输出数据为所述Ni标号组F函数输出信息。其中，异或操作 (XOR_Operation)的具体步骤如下：

[0143] 步骤4.4.1:将输入的数据$X_{bs}[0]$(或$X_{bs}[2]$)从高位到低位按4位一组划分成4个部分，从高位到低位按4位一组分别依次记作$xor_0$、$xor_1$、$xor_2$及$xor_3$；

[0144] 步骤4.4.2:将$xor_0$、$xor_3$分别对应与$xor_2$、$xor_1$进行异或操作得到$xor_1$'和$xor_2$'，即 $xor_1'=xor_0 \oplus xor_2$， $xor_2'=xor_3 \oplus xor_1$；

[0145] 步骤4.4.3:将步骤4.4.2得到的$xor_1$'、$xor_2$'分别对应与$xor_0$、$xor_3$进行异或操作得到$xor_0$' 和$xor_3$'，即 $xor_0'=xor_1' \oplus xor_0$， $xor_3'=xor_2' \oplus xor_3$；

[0146] 步骤4.4.4:将步骤4.4.2和4.4.3得到的结果按从左至右从高位到低位的顺序排列为$xor_0$'、 $xor_1$'、$xor_2$'和$xor_3$'，将它们连接并输出数据得到$X_f[0]$或($X_f[2]$)。

[0147] 应当理解，按照步骤4.4.1-4.4.4分别处理$X_{bs}[0]$、$X_{bs}[2]$得到$X_f[0]$、$X_f[2]$。

[0148] 关于FF函数操作部分，在进入F函数操作的同时也进入FF函数进行操作。FF函数是简单的二类广义Feistel网络结构，主要由S盒和循环左移操作组成，其中S盒记作sbox。FF函数的轮数与外部轮运算轮数构成一个负反馈机制。FF函数的轮数按照如下规则确定：

[0149] 若轮数R满足：15≤R<20，所述FF函数的轮数为8轮；

[0150] 若轮数R满足：20≤R<25，所述FF函数的轮数为6轮；

[0151] 若轮数R满足：25≤R≤30，所述FF函数的轮数为4轮。

[0152] 本发明输入数据$X_1[0]$、$X_1[2]$进入FF函数进行操作之后得到$X_{ff}[0]$、$X_{ff}[2]$数据。FF函数中，S盒引用用Gift算法加密的S盒，如表3所示：

[0153] 表3 sbox盒元素

[0154]

| x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| S[x] | 1 | A | 4 | C | 6 | F | 3 | 9 | 2 | D | B | 7 | 5 | 0 | 8 | E |

[0155] 关于按位异或操作部分：

[0156] 将F函数输出得到的数据$X_f[0]$、$X_f[2]$又分别对应与X[1]、X[3]以及当前轮运算对应的外部轮密钥进行按位异或操作对应得到X'[1]和X'[3]，即$X'[1]=X_f[0]\oplus X[1]\oplus rk_{2i}$，$X'[3]=X_f[2]\oplus X[3]\oplus rk_{2i+1}$，$rk_{2i}$，$rk_{2i+1}$分别表示当前轮数i对应的两个外部轮密钥。i的取值为0至R-1，本发明得到的外部轮密钥的个数为2R个，表示为：$rk_0$，$rk_1$....$rk_{2R-1}$。

[0157] 公式c为密钥长度为96位时外部轮密钥计算公式，公式d为密钥长度为128位时外部轮密钥计算公式：

[0158] 公式c：

[0159] $$(rk_{2i},rk_{2i+1})\leftarrow(con_{2i}^{96},con_{2i+1}^{96})\oplus\begin{cases}(k\_i[2],k\_i[3]) & if \quad i \bmod 5=0 or 2\\(k\_i[0],k\_i[1]) & if \quad i \bmod 5=1 or 4\\(k\_i[4],k\_i[4]) & if \quad i \bmod 5=3\end{cases}$$

[0160] $$(con_{2i}^{96}\,|\,con_{2i+1}^{96})\leftarrow(c_{i+1}\,|\,c_0\,|\,c_{i+1}\,|\,\{00\}_2\,|\,c_{i+1}\,|\,c_0\,|\,c_{i+1}\,|)\oplus\{a96cd4b8\}_{16}$$

[0161] 公式d：

[0162] $$\left.\begin{aligned}rk_{2i}&\leftarrow k\_i[(2i+2)\bmod 8]\oplus con_{2i}^{128}\\rk_{2i+1}&\leftarrow k\_i[(2i+3)\bmod 8]\oplus con_{2i+1}^{128}\end{aligned}\right\}$$

[0163] $$(con_{2i}^{128}\,|\,con_{2i+1}^{128})\leftarrow\left(c_{i+1}|c_0|c_{i+1}|\{00\}_2|c_{i+1}|c_0|c_{i+1}|\right)\oplus\{8b79a465\}_{16}$$

[0164] 式中，$rk_{2i}$，$rk_{2i+1}$表示当前轮数对应的轮密钥，|为连接符，⊕为异或符，mod代表取余运算，$\{\}_{16}$为十六进制数，$\{\}_2$为二进制数，i为当前轮数，$c_i$是用5位二进制数表示的十进制数i，例如，$c_0=\{00000\}_2$，$c_{11}=\{01011\}_2$。

[0165] 将上述按位异或操作得到的数据X'[1]和X'[3]以及FF函数操作得到的数据$X_{ff}[0]$、$X_{ff}[2]$组成X'[1]、$X_{ff}[2]$、X'[3]和$X_{ff}[0]$。

[0166] 关于下一轮轮运算，若当前轮数i小于R轮，则重复上述步骤进行轮运算，下一轮的轮运算的4组加密信息分别是上一轮的输出结果。即将当前轮运算的数据X'[1]、$X_{ff}[2]$、X'[3] 和$X_{ff}[0]$作为下一轮轮运算的加密信息$X_1[0]$、X[1]、$X_1[2]$和X[3]。

[0167] S5：利用步骤S3中剩余白化密钥$wk_2$、$wk_3$对步骤S4中到4组输出数据中$X_{ff}[0]$、$X_{ff}$

[2] 组对应数据进行异或处理得到X'ff[0]、X'ff[2]，最终得到4个数据块X'[1]、X'ff[2]、X'[3]、X'ff[0]。

[0168] S6：将步骤S5中4个数据块按照X'[1]、X'ff[2]、X'[3]、X'ff[0]的顺序作为Sa置换操作的初始值进行Sa置换操作得到输出数据，并将输出数据与初始加密密钥的低位进行轮加密操作。

[0169] Sa置换运算流程如图7 所示，是将输入的64位数据从高位按16位一组划分成4×4的数据块矩阵，记为P1、P2、P3、P4，分别对这四个数据块矩阵按照置换表$Pa_1$、$Pa_2$、$Pa_3$、$Pa_4$进行置换。

[0170] 表4 Whee1算法置换表$Pa_1$元素

[0171]

| $Pa_1$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| P | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $Pa_1(i)$ | 2 | 3 | 7 | 6 | 10 | 11 | 15 | 14 |
| P | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| $Pa_1(i)$ | 13 | 12 | 8 | 9 | 5 | 4 | 0 | 1 |

[0172] 表5 Whee1算法置换表$Pa_2$元素

[0173]

| $Pa_2$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| P | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $Pa_2(i)$ | 8 | 12 | 13 | 9 | 10 | 14 | 15 | 11 |
| P | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| $Pa_2(i)$ | 3 | 7 | 6 | 2 | 1 | 5 | 4 | 0 |

[0174] 表6 Whee1算法置换表$Pa_3$元素

[0175]

| $Pa_3$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| P | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $Pa_3(i)$ | 3 | 2 | 6 | 7 | 11 | 10 | 14 | 15 |
| P | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| $Pa_3(i)$ | 0 | 1 | 5 | 4 | 8 | 9 | 13 | 12 |

[0176] 表7 Whee1算法置换表$Pa_4$元素

[0177]

| $Pa_4$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| P | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| $Pa_4(i)$ | 15 | 11 | 10 | 14 | 13 | 9 | 8 | 12 |
| P | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| $Pa_4(i)$ | 0 | 4 | 5 | 1 | 2 | 6 | 7 | 3 |

[0178] 具体的，如图7 所示，Sa置换操作的过程如下：

[0179] S6.1：将4个数据块从左至右标记为$P_1$、$P_2$、$P_3$和$P_4$，并分别对所述4个数据块从高位到低位依次划分为8个字节$g_0$，$g_1$，$g_2$，$g_3$，$g_4$，$g_5$，$g_6$，$g_7$；

[0180] 每个数据块从高位到低位依次划分为2个字节，每个数据块均为4×4矩阵；

[0181] 数据块$P_1$从高位到低位依次划分为8个字节$g_0$，$g_1$，数据块$P_2$从高位到低位依次划分为 8个字节$g_2$，$g_3$，数据块$P_3$从高位到低位依次划分为8个字节$g_4$，$g_5$，数据块$P_4$从高位到低位依次划分为8个字节$g_6$，$g_7$；

[0182] S6.2：分别将4数据块$P_1$、$P_2$、$P_3$和$P_4$依数据块矩阵的两条中心对称轴划分成4个2×2 的数据块矩阵，并从左上角的2×2数据块矩阵开始按照顺时针顺序分别对应给所述4个2

×2 的数据块矩阵编号为$A_i$、$B_i$、$D_i$和$C_i$(1≤i≤4)，以及对每个2×2的数据块矩阵$A_i$、$B_i$、$D_i$和 $C_i$从左上角开始按照顺时针对应编号为$a_1$、$a_2$、$a_4$和$a_3$，$b_1$、$b_2$、$b_4$和$b_3$，$d_1$、$d_2$、$d_4$和$d_3$以及$c_1$、$c_2$、$c_4$和$c_3$；

[0183] S6.3：对每个数据块中的4个2×2数据块矩阵$A_i$、$B_i$、$D_i$和$C_i$分别按照预设顺序进行置换操作，过程如下：

[0184] 对于数据块$P_1$，将数据块矩阵$P_1$中$A_1$区域从$a_2$开始按逆时针连接到$a_4$结束组成$g_0$'的前半字节，接着$C_1$区域也从$c_2$开始按逆时针连接到$c_4$结束组成$g_0$'的后半字节，$B_1$区域从$b_1$开始按顺时针连接到$b_3$结束组成$g_1$'的前半字节，$D_1$区域也从$d_1$开始按顺时针连接到$d_3$结束组成$g_1$'的后半字节；

[0185] 对于数据块$P_2$，将数据块矩阵$P_2$中$B_2$区域从$b_4$开始按逆时针连接到$b_3$结束组成$g_2$'的前半字节，接着$A_2$区域也从$a_4$开始按逆时针连接到$a_3$结束组成$g_2$'的后半字节，$C_2$区域从$c_1$开始按逆时针连接到$c_2$结束组成$g_3$'的前半字节，$D_2$区域也从$d_1$开始按逆针连接到$d_2$结束组成$g_3$'的后半字节；

[0186] 对于数据块$P_3$，将数据块矩阵$P_3$中$C_3$区域从$c_3$开始按逆时针连接到$c_1$结束组成$g_4$'的前半字节，接着$A_3$区域也从$a_3$开始按逆时针连接到$a_1$结束组成$g_4$'的后半字节，$D_3$区域从$d_4$开始按顺时针连接到$d_2$结束组成$g_5$'的前半字节，$B_3$区域也从$b_4$开始按逆顺针连接到$b_2$结束组成$g_5$'的后半字节；

[0187] 对于数据块$P_4$，将数据块矩阵$P_4$中$A_4$区域从$a_1$开始按逆时针连接到$a_2$结束组成$g_6$'的前半字节，接着$B_4$区域也从$b_1$开始按逆时针连接到$b_2$结束组成$g_6$'的后半字节，$D_4$区域从$d_4$开始按逆时针连接到$d_3$结束组成$g_7$'的前半字节，$C_4$区域也从$c_4$开始按逆顺针连接到$c_3$结束组成$g_7$'的后半字节；

[0188] S6.4：按照$g_3$'，$g_7$'，$g_5$'，$g_6$'，$g_1$'，$g_4$'，$g_0$'，$g_2$'的顺序进行连接并输出数据形成 64位数据输出。

[0189] 最后是将得到的64位输出数据，将其与初始密钥中低64位进行轮密钥加操作，得到加密结果并输出。

[0190] 本发明Wheel算法测试向量如表8、表9所示：

[0191] 表8 Wheel-96测试数据

[0192]

| Plaintext | Key | Ciphertext |
|---|---|---|
| 0000_0000_0000_0000 | 0000_0000_0000_0000_0000_0000 | EC64_EC4A_CF8F_EDC3 |
| FFFF_FFFF_FFFF_FFFF | FFFF_FFFF_FFFF_FFFF_FFFF_FFFF | 207B_2282_017F_10D0 |
| 0000_0000_0000_0000 | FFFF_FFFF_FFFF_FFFF_FFFF_FFFF | 139C_3049_32C0_2103 |
| 0123_4567_89AB_CDEF | 0123_4567_89AB_CDEF_0123_4567 | FC66_9A60_7795_031E |

[0193] 表9 Wheel-128测试数据

[0194]

| Plaintext | Key | Ciphertext |
|---|---|---|
| 0000_0000_0000_0000 | 0000_0000_0000_0000_0000_0000_0000_0000 | DF03_FE71_EDA6_DDAB |
| FFFF_FFFF_FFFF_FFFF | FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF | 31EB_30D0_037C_1230 |
| 0000_0000_0000_0000 | FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF_FFFF | 0326_3332_12D7_314E |
| 0123_4567_89AB_CDEF | 0123_4567_89AB_CDEF_0123_4567_89AB_CDEF | CECB_BB48_7791_32B7 |

[0195] 本发明所述的Wheel-96密码算法在Modelsim SE 6.1f Evaluation上进行仿真；在 SynopsysDesignComiler Version B-2008.09进行综合，其中综合工艺库为SMIC 0.18μm CMOS，在综合实验中，面积资源用等效门数GE来衡量。

[0196] Wheel算法各组件硬件实现资源具体描述：64位明文保存在寄存器中需要344GE，96位保存在寄存器中需要580.8GE。一次加密时，F函数中密钥与明文的16位异或运算，需要38GE，因此需要38*2＝76GE；F函数中S盒替换层使用的S盒是PRESENT和Piccolo算法的S盒，需要192.66GE；F函数中XOR操作需要32GE。FF函数中S盒替换层使用的GIFT的S盒，需要24GE；FF函数是基于Feistel网络结构，与外部轮运算构成负反馈机制，因此轮数有4 轮、6轮和8轮，其中每一轮有两个异或操作，因此当轮数为4轮时，需要16GE；当轮数为 6轮时，需要24GE；当轮数为8轮时，需要32GE。FF函数中Bit置换和Sa置换，采用连线方式实现，硬件实现不需要消耗资源。最后一个轮密钥加操作64位密钥和明文的异或操作，需要172GE。Wheel-96算法硬件实现最多仅需要1453.46GE。表10是Wheel-96算法ASIC 资源面积列表。

[0197] 表10 Wheel-96算法资源列表

[0198]

| 算法模块 | GE |
|---|---|
| 明文寄存器 | 344 |
| 密钥寄存器 | 580.8 |
| 64位异或单元 | 172 |
| 16位异或单元 | 76 |
| F函数中S盒替换层 | 192.66 |
| F函数中XOR操作 | 32 |
| FF函数中S盒替换层 | 24 |
| FF函数中4/6/8轮异或操作 | 16/24/32 |
| Sa置换/F函数中bit置换 | 0 |
| 总和(最大) | 1453.46 |

[0199] 满足不同用户多层次的高效率需求，采用两种密钥长度。算法采用广义Feistel结构，通过密钥高8位来控制轮数的变换，在确保算法安全性的前提下，轮数在一定范围内变动，在一次加密过程中适当的减少算法的轮数，这样能够有效的提高算法的效率。轮运算结束后进入Sa置换操作从而提高扩散性。综上是的算法具有灵活性、高效性和低资源消耗的特点，相比于其它基于Feistel结构的轻量级算法安全和加密性能更加优越。

[0200] 表11各分组密码算法ASIC实现

[0201]

| 算法 | 结构 | 分组长度 | 密钥长度(bits) | 资源面积(GE) |
|---|---|---|---|---|
| PRESENT-80 | SPN | 64 | 80 | 1570 |
| Twine-128 | Feistel | 64 | 128 | 1866 |
| Piccolo-128 | GFN | 64 | 128 | 1938 |
| CLEFIA | GFN | 64 | 128 | 2488 |
| mCRYPTON | SPN | 64 | 128 | 2500 |
| Cube Cipher | SPN | 64 | 128 | 2536 |
| PUFFIN | SPN | 64 | 128 | 2577 |
| Wheel-96 | GFN | 64 | 96 | 1453.46 |
| Wheel-128 | GFN | 64 | 128 | 1640.66 |

[0202] 需要强调的是,本发明所述的实例是说明性的,而不是限定性的,因此本发明不限于具体实施方式中所述的实例,凡是由本领域技术人员根据本发明的技术方案得出的其他实施方式,不脱离本发明宗旨和范围的,不论是修改还是替换,同样属于本发明的保护范围。

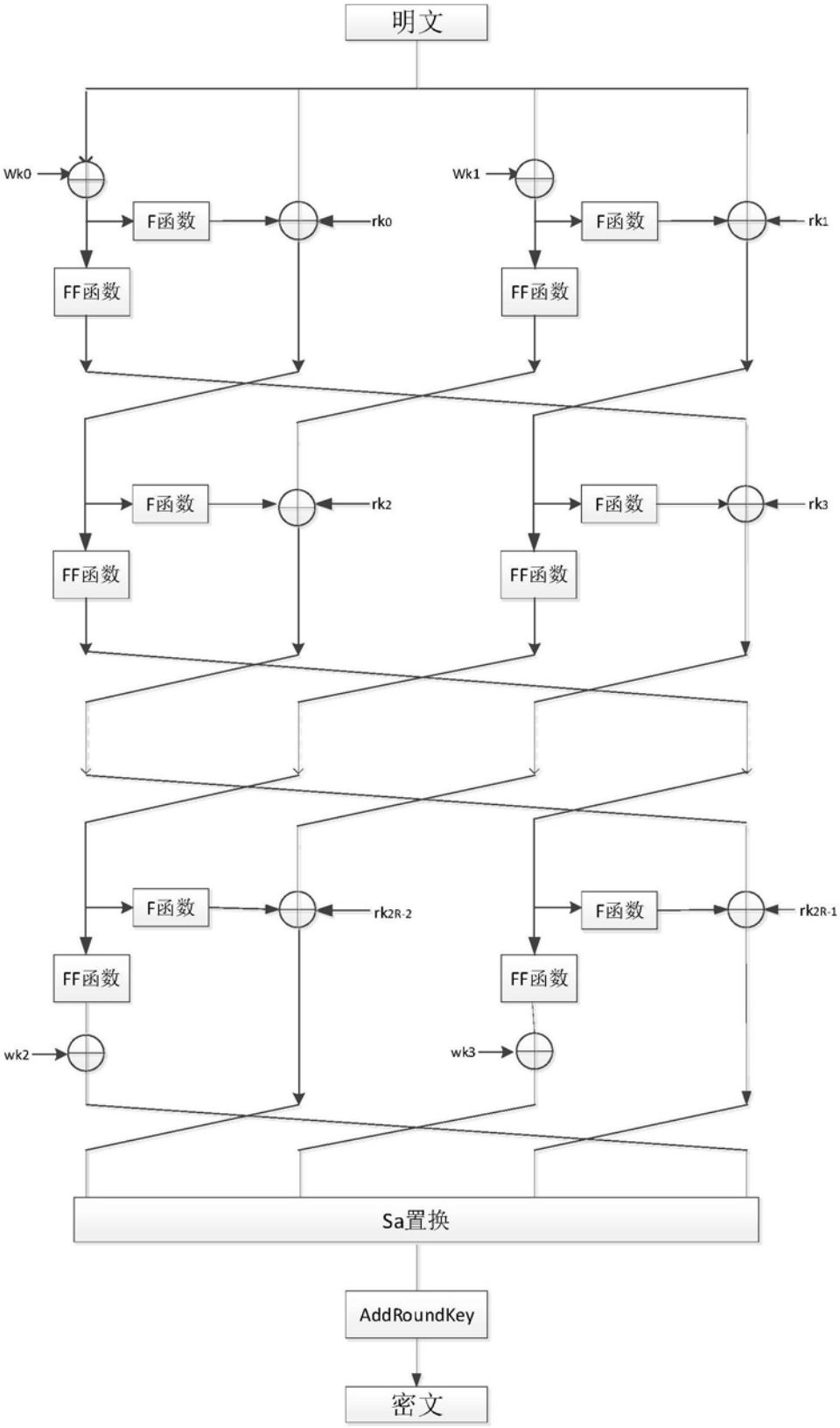

明文
Wk0
Wk1
F函数
rk0
rk1
FF函数
rk2
rk3
rk2R-2
rk2R-1
wk2
wk3
Sa置换
AddRoundKey
密文

图1

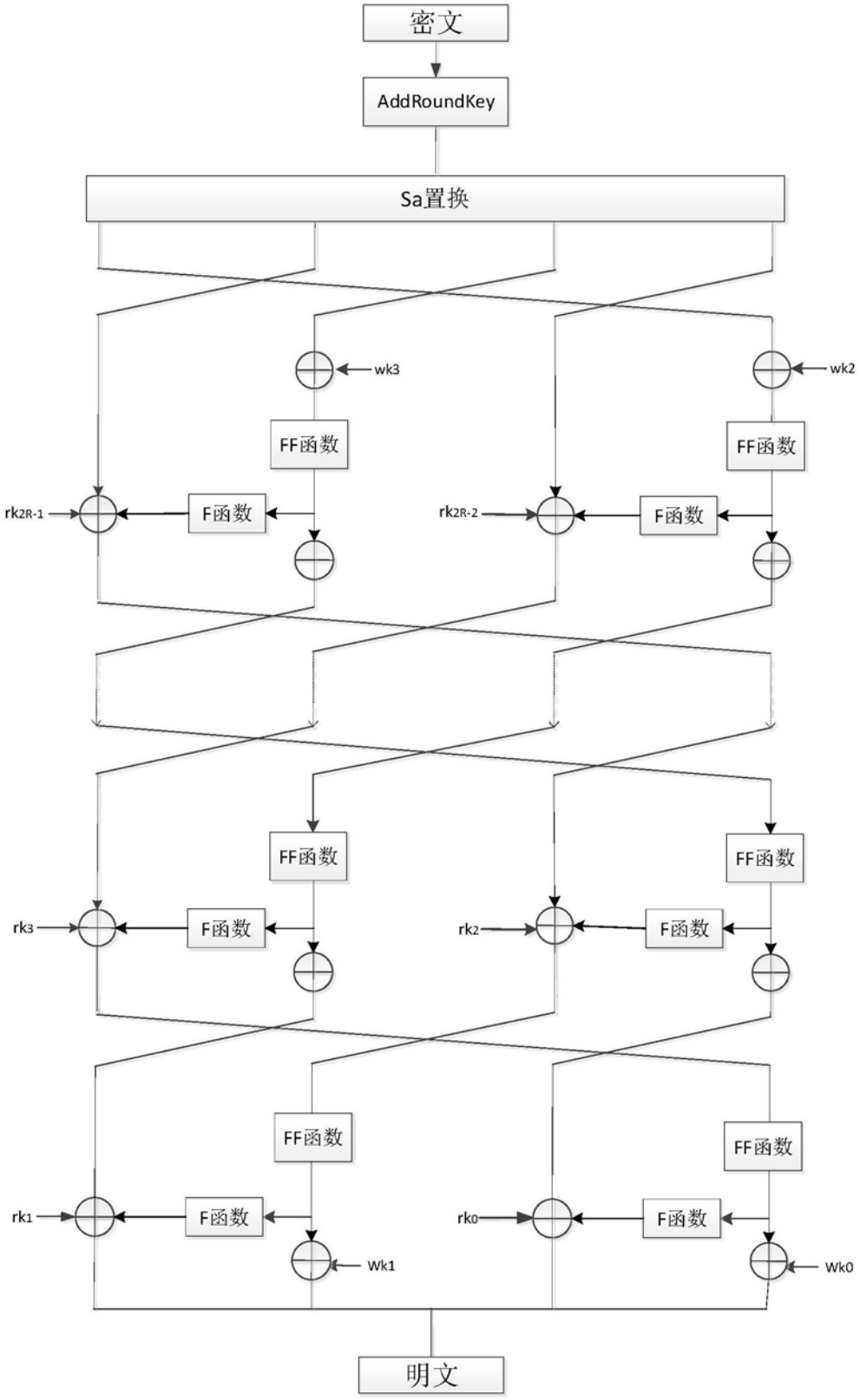

密文
AddRoundKey
Sa置换
wk3
wk2
FF函数
FF函数
rk2R-1
F函数
rk2R-2
F函数
FF函数
FF函数
rk3
F函数
rk2
F函数
FF函数
FF函数
rk1
F函数
rk0
F函数
Wk1
Wk0
明文

图2

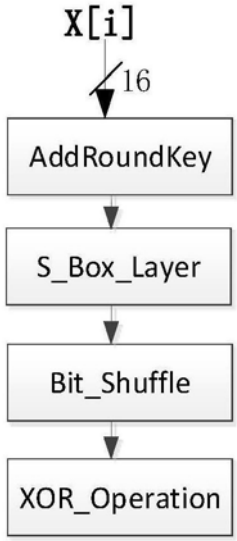

X[i]
16
AddRoundKey
S_Box_Layer
Bit_Shuffle
XOR_Operation

图3

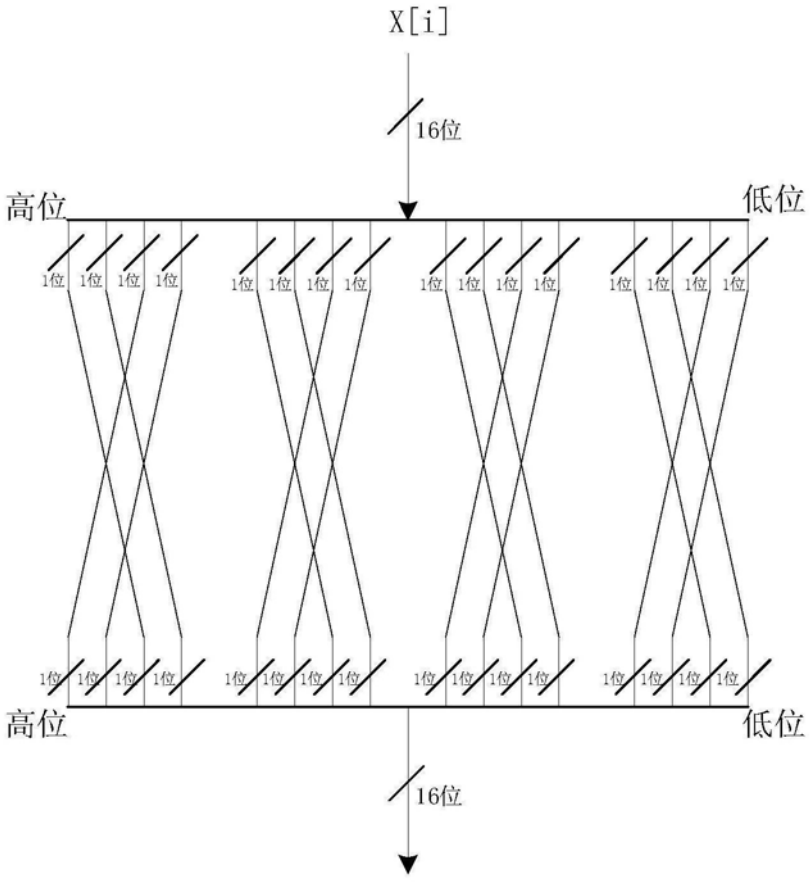

X[i]
16位
高位
低位
1位
高位
低位
16位

图4

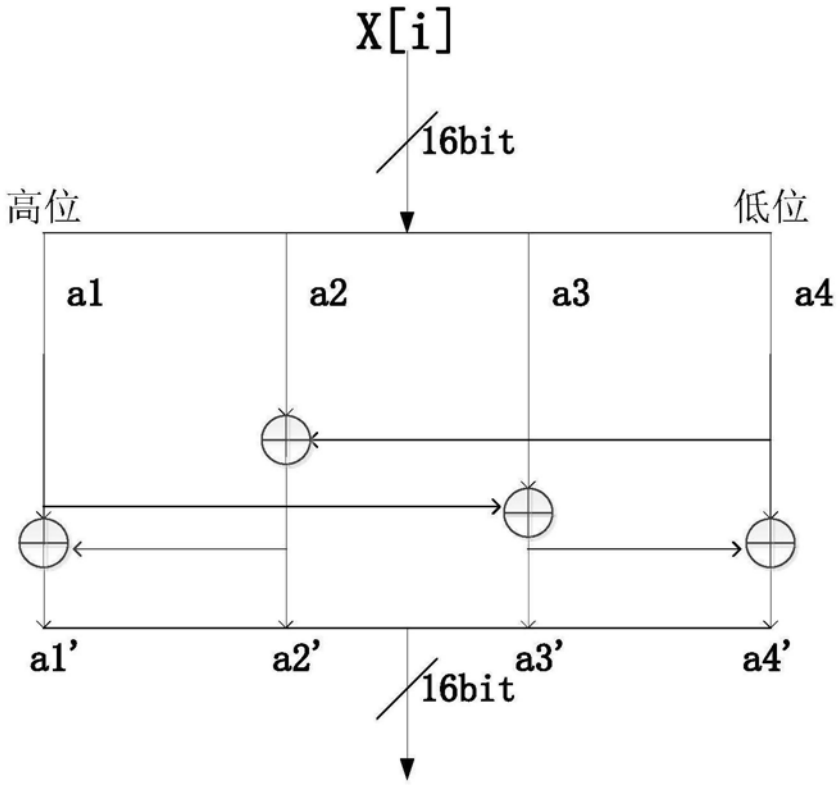



图5

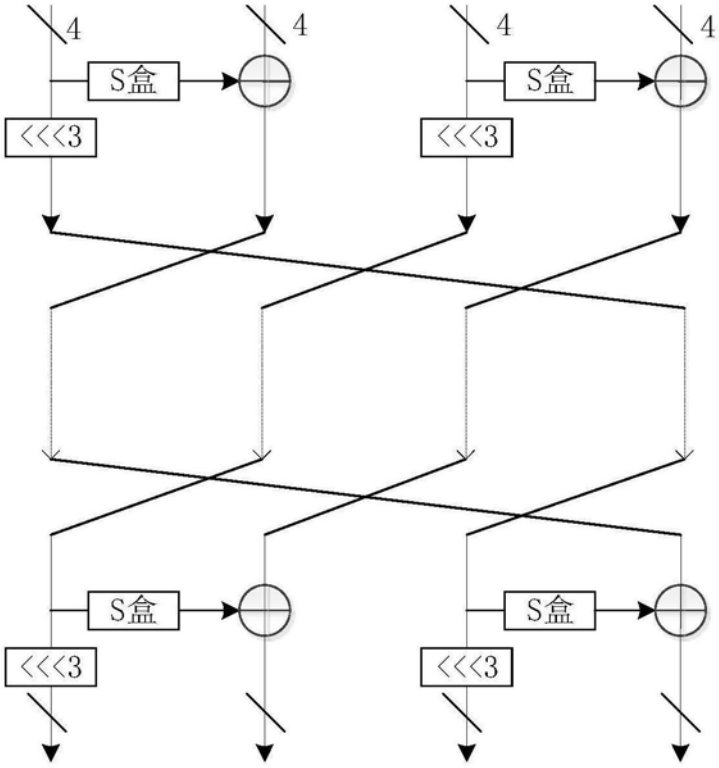



图6

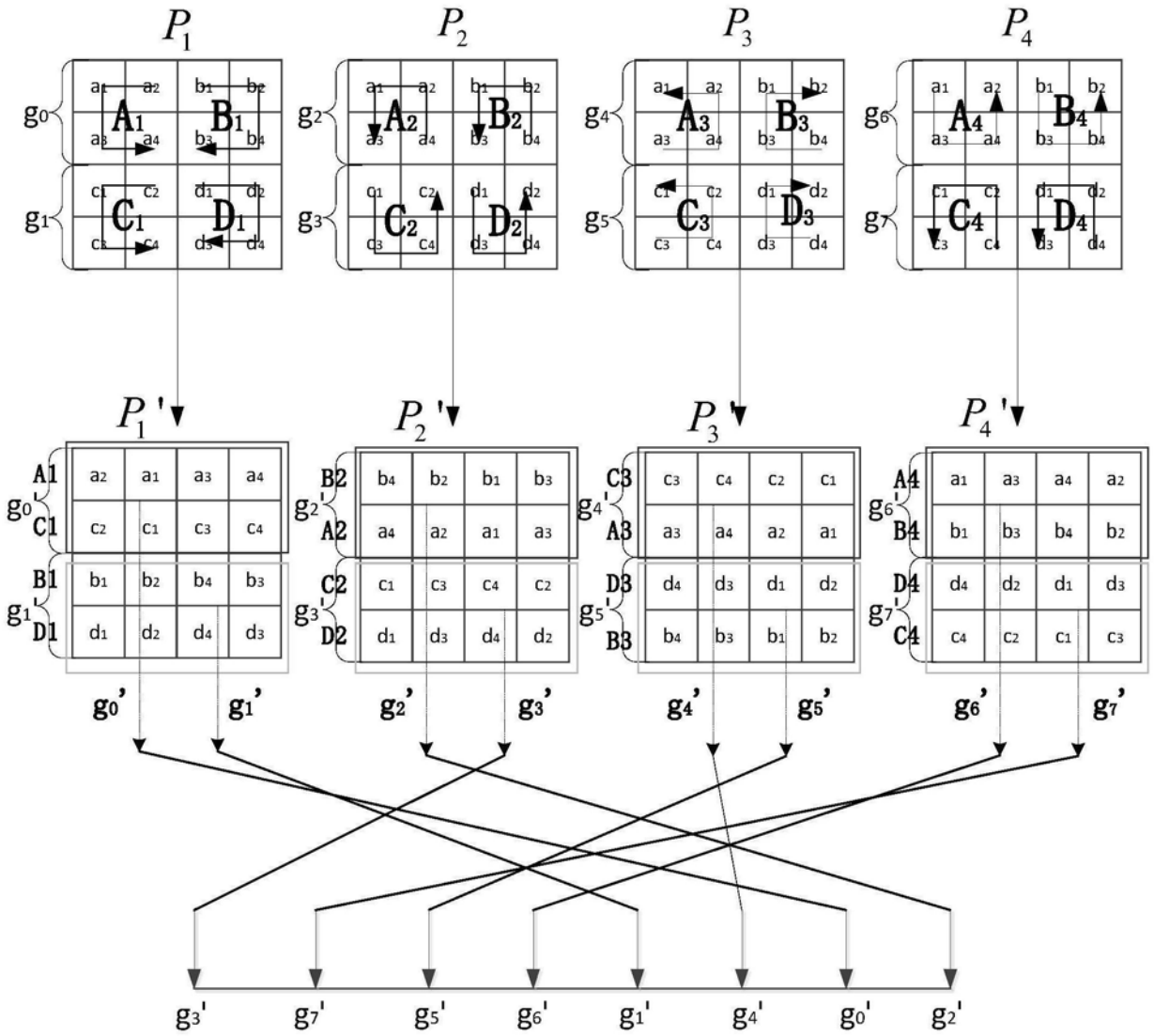

P1
P2
P3
P4
g0
g1
g2
g3
g4
g5
g6
g7
A1
B1
C1
D1
A2
B2
C2
D2
A3
B3
C3
D3
A4
B4
C4
D4
P1'
P2'
P3'
P4'
g0'
g1'
g2'
g3'
g4'
g5'
g6'
g7'
g3' g7' g5' g6' g1' g4' g0' g2'

图7