基于Feistel结构的超轻量级分组密码算法(PFP)

黄玉划[1,3] 代学俊[1] 时阳阳[1] 刘宁钟[1] 曾庆喜[2] 苏 菲[3]
(南京航空航天大学计算机科学与技术学院 南京 211106)[1]
(南京航空航天大学能源与动力学院 南京 210016)[2] (苏州中科启慧软件技术有限公司 苏州 215500)[3]

摘 要 面向无线终端资源受限环境对加密算法的应用需求,借鉴PRESENT算法的设计思想,采用Feistel结构,并修改扩散层的P置换,设计了一种超轻量级分组密码算法PFP。其硬件实现需要1355GE,优于PRESENT算法,满足资源极端受限环境的需求(2000GE以下)。速度测试结果表明,PFP算法的软件实现效率约为PRESENT算法的1.5倍。依赖性测试、线性分析、差分分析、不可能差分分析和密钥编排攻击表明,PFP算法满足轻量级分组密码的安全需求。

关键词 轻量级分组密码,Feistel结构,PRESENT算法,依赖性测试,密码分析
中图法分类号 TP309,TN918 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.03.036

Ultra-lightweight Block Cipher Algorithm (PFP) Based on Feistel Structure

HUANG Yu-hua[1,3] DAI Xue-jun[1] SHI Yang-yang[1] LIU Ning-zhong[1] ZENG Qing-xi[2] SU Fei[3]
(College of Computer Science and Technology, Nanjing University of Aeronautics & Astronautics, Nanjing 211106, China)[1]
(College of Energy and Power Engineering, Nanjing University of Aeronautics & Astronautics, Nanjing 210016, China)[2]
(Suzhou Chinsdom Co. Ltd., Suzhou 215500, China)[3]

Abstract To meet the application requirement for cipher algorithms in the resource-constrained terminal system such as the limited energy supply etc, an ultra-lightweight block cipher named PFP was designed by using the experience of PRESENT algorithm for reference. The iterative structure of PFP algorithm is Feistel structure. Its permutation was modified in diffusion layer. It requires only 1355GE with hardware implementation of PFP algorithm, which is better than the PRESENT. And it also fulfills the requirement of environment with extremely constrained resource (no more than 2000GE). Test results show that the speed of PFP algorithm is about 50% faster than PRESENT. Depen-dence test, linear analysis, differential analysis, impossible differential analysis and key schedule attack show that the PFP algorithm can satisfy the security requirements of the lightweight block cipher algorithm.

Keywords Lightweight block cipher, Feistel structure, PRESENT algorithm, Dependence test, Cryptanalysis

1 引言

分组密码算法是密码学中的一个研究热点,被广泛应用于网络与信息系统安全中的数据加解密。分组密码还可用于设计流密码算法、Hash函数与MAC算法,从而实现数据校验与消息认证、身份认证与密钥交换、伪随机数产生及数字签名[1-4]。随着无线通信与网络技术的发展,普通的分组密码算法难以满足无线终端资源受限环境的应用需求,轻量级分组密码的设计与分析成为了当前的研究热点,一大批轻量级分组密码算法[5-6]被提出。

目前,设计轻量级分组密码的方法主要有两种:1)在现有密码算法的基础上,对密码算法的组件进行轻量化的改进;2)从零开始,设计一种新的轻量级密码算法。第一种方法是借助已有算法的安全性和健壮性,在尽可能不降低安全性甚至是提高安全性的前提下使算法轻量化。这种方法的优点是设计工作量小,安全性分析也因基于原有算法的分析而相对容易。而这种方法的缺点有:1)改进算法实现所需要的资源时受到原有算法结构的限制,因此应该选择硬件实现效率高的算法;2)在原有算法上所做的修改可能会使算法出现一些新的弱点,所以采用这种方法设计的轻量级算法存在遭遇新型攻击的可能。第二种方法没有固定算法的限制,设计比较灵活,设计者可以很好地利用可行的方式使算法满足资源受限的要求,只是采用这种方法设计出的算法要进行很多安全性分析。

分组密码算法的设计结构主要有SP,Feistel和广义Feistel结构等[4]。轻量级分组密码国际标准PRESENT算

到稿日期:2015-11-06 返修日期:2016-03-07 本文受江苏省科技支撑计划项目(BE2013879),国家自然科学基金项目(61375021),江苏省自然科学基金项目(SBK201322136),南京航空航天大学青年科技创新基金项目(NS2010097)资助。

黄玉划(1975-),男,博士后,副教授,CCF会员,主要研究方向为无线通信及其安全技术,E-mail:hyuhua2k@163.com;**代学俊**(1991-),女,硕士生,主要研究方向为信息安全;**时阳阳**(1986-),女,硕士,主要研究方向为信息安全;**刘宁钟**(1975-),男,博士,教授,博士生导师,主要研究方向为嵌入式系统;**曾庆喜**(1980-),男,博士后,讲师,主要研究方向为嵌入式系统;**苏 菲**(1977-),女,硕士,主要研究方向为嵌入式系统。

法[6-8]的设计采用 SP 结构。Feistel 结构本身是可逆的，不需要专门设计解密算法，这样可以减小内存需求并提高硬件实现效率，一般更适合轻量级分组密码算法设计。本文采用 Feistel 结构来设计轻量级分组密码算法。

2 PRESENT 算法介绍

Bogdanov 等在 CHES 2007 上提出了 PRESENT 算法，该算法是一个超轻量级分组密码。PRESENT 算法采用 SP 网络结构，分组长度为 64bits，加密轮数为 31 轮，密钥长度分 80bits 和 128bits 两种。设计者认为 80bits 密钥已经能够保证安全性，所以推荐使用 80bits 密钥长度。每轮加密过程中，采用异或运算引入轮密钥 K_i($1\leqslant i\leqslant 32$)。算法中的非线性置换使用 16 个相同的 4 位 S 盒并行，置换层用到一个 P 置换。S 盒如表 1 所列。整个算法的加密流程如图 1 所示。

表 1 PRESENT 算法的 S 盒

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

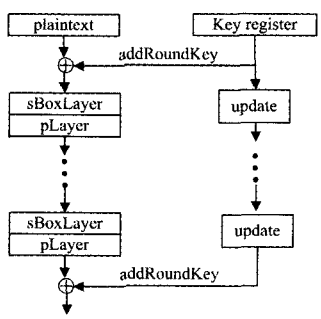

图 1 PRESENT 算法加密流程图

PRESENT 加解密全过程可分为以下 3 部分。

2.1 密钥扩展(对应图 1 的右部)

以 80bit 密钥的 PRESENT 密码为例，其方法和过程可类推到 128bit 的 PRESENT 密码中。假设密钥 $K=k_{79}k_{78}\cdots k_0$，轮密钥表示为 $K_i=k_{63}^ik_{62}^i\cdots k_0^i$，轮密钥扩展进行 32 轮的相同迭代变换，轮变换的输入、输出均为 80bit，每一轮的轮密钥都取相应轮次密钥扩展变换之后输出的 80bit 中最左侧的 64bit。轮密钥扩展算法如下：

(1) $[k_{79}k_{78}\cdots k_0]=[k_{18}k_{17}\cdots k_{20}k_{19}]$(循环右移 18 位)；

(2) $[k_{79}k_{78}k_{77}k_{76}]=S[k_{79}k_{78}k_{77}k_{76}]$；

(3) $[k_{19}k_{18}k_{17}k_{16}k_{15}]=[k_{19}k_{18}k_{17}k_{16}k_{15}]\oplus rc$。

其中，rc 为轮次数，即 $0\leqslant rc\leqslant 31$；$S[k]$表示用 S 盒进行变换。

2.2 迭代加密(对应图 1 的左部)

(1)31 轮迭代每轮加密过程($1\leqslant i\leqslant 31$)分为以下 3 步。

1)轮密钥异或(AddRoundKey)：设轮密钥为 $K_i=k_{63}^ik_{62}^i\cdots k_0^i$，当前状态为 $B=b_{63}\cdots b_0$，则变换如下：

$$B=B\oplus K_i$$

2)S 盒代换(sBoxLayer)：设状态为 $B=b_{63}\cdots b_0$，可表示成 16 个 4bit 数 $w_i=b_{4i+3}b_{4i+2}b_{4i+1}b_{4i}$，其中 $0\leqslant i\leqslant 15$。则采用表 1 中的 S 盒进行如下变换：

$$w_i=S[w_i]\ (0\leqslant i\leqslant 15)$$

3)P 置换(pLayer)：P 置换将 64bit 状态进行重排，提供扩散效应。PRESENT 算法的 P 置换函数为：

$$P(i)=\begin{cases}i\cdot 16 \bmod 63, & 0\leqslant i\leqslant 62\\ 63, & i=63\end{cases}$$

由此得到的 P 置换表如表 2 所列。

表 2 PRESENT 算法的 P 置换表

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	16	32	48	1	17	33	49	2	18	34	50	3	19	35	51
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	20	36	52	5	21	37	53	6	22	38	54	7	23	39	55
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	8	24	40	56	9	25	41	57	10	26	42	58	11	27	43	59
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	12	28	44	60	13	29	45	61	14	30	46	62	15	31	47	63

(2)在最后一轮进行一次密钥白化过程即轮密钥异或(Add-RoundKey)，得到最终的密文 C。

2.3 解密算法

PRESENT 密码解密过程与加密过程不是同一个算法，解密过程中需要对每个加密部件进行取逆运算并进行 31 轮的解密轮函数运算，同时还包括最后的密钥白化过程。其中，密钥的使用与加密过程正好相反。具体操作如下。

(1)31 轮迭代每轮解密过程($1\leqslant i\leqslant 31$)分为以下 3 步。

1)逆轮密钥异或(InvAddRoundKey)：反向采用轮密钥进行异或运算。

2)逆 P 置换(InvpLayer)：将解密模块中 64bit 进行重排，具体如表 3 所列。

表 3 PRESENT 算法的逆 P 置换表

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	1	5	9	13	17	21	25	29	33	37	41	45	49	53	57	61
i	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
$P(i)$	2	6	10	14	18	22	26	30	34	38	42	46	50	54	58	62
i	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
$P(i)$	3	7	11	15	19	23	27	31	35	39	43	47	51	55	59	63

3)逆 S 盒代换(InvsBoxLayer)：逆 S 盒代换可以通过查表进行，如表 4 所列。

表 4　PRESENT 算法的逆 S 盒

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

(2) 在最后一轮再与初始轮密钥进行异或(InvAddRoundKey)，从而得到最终的明文。

3　基于 Feistel 结构的轻量级分组密码(PFP)

由于 Feistel 结构本身是可逆的，不需要专门设计解密算法，这样可以减小内存需求并提高硬件实现效率。我们采用 Feistel 结构，借鉴 PRESENT 算法，并修改扩散层的 P 置换，得到 PFP 算法。PFP 算法的分组长度为 64bits，密钥长度为 80bits(密钥长度为 128bits 时方法和过程与此相似)，迭代轮数选择为 34，加密时 64bits 的分组数据分为左、右两个分支 L_i 和 R_i。每轮加密更新一个分支，轮函数由线性置换、非线性置换和子密钥加组成，其中轮函数的设计借鉴 PRESENT 算法。

输入的 64bits 分组明文分为左、右两半即 $L_0 \| R_0$，L_0 和 R_0 长度都是 32bits，PFP 算法通过如下计算输出密文 $L_{34} \| R_{34}$。

$L_i = R_{i-1}$；$R_i = L_{i-1} \oplus F(R_{i-1}, K_{i-1})$

其加密流程如图 2 所示。

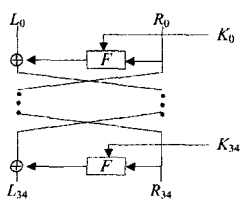

图 2　PFP 算法加密流程

3.1　子密钥加

子密钥 $K_i = k_{31}^i k_{30}^i \cdots k_0^i$ $(0 \leqslant i \leqslant 34)$ 按顺序取自扩展密钥。密钥扩展算法将在下文介绍。子密钥加指简单地将一个子密钥按位异或到右分支 R_i 上，即 $R_i = b_{31} \cdots b_0$，则变换如下：$R_i = R_i \oplus K_i$。

3.2　S 盒的设计

轻量级分组密码的设计要兼顾硬件的实现效率，所以一个 4×4 的 S 盒要比 8×8 的 S 盒更简单、紧凑。S 盒是一个非线性变换，它将加密过程中的每个字节非线性地转化为另一个字节。PFP 算法中轮函数 F 的非线性变换采用 PRESENT 算法的 S 盒，加密时 8 个相同的 S 盒并行使用。

$f: GF(2^{32}) \rightarrow GF(2^{32})$

$(b_1, b_2, b_3, b_4, b_5, b_6, b_7, b_8) = f(a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8)$

$b_i = S(a_i)(1 \leqslant i \leqslant 8)$

3.3　置换层设计

为了使算法具有更好的硬件实现性能，在设计置换层时，没有选择类似 AES 算法的置换层设计方法，而选择了位排列置换。PFP 算法中的 P 置换表如表 5 所列，即数据经过 P 置换后第 i 位移动到第 $P(i)$ 位。也可以把置换层用公式表示为：

$$P(i) = \begin{cases} i \cdot 8 \bmod 31, & 0 \leqslant i \leqslant 30 \\ 31, & i = 31 \end{cases}$$

表 5　PFP 算法的 P 置换表

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$P(i)$	0	8	16	24	1	9	17	25	2	10	18	26	3	11	19	27
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
$P(i)$	4	12	20	28	5	13	21	29	6	14	22	30	7	15	23	31

3.4　密钥扩展算法

80bits 的原始密钥存储在一个密钥寄存器 K 中，可以表示为 $K = k_{79} k_{78} \cdots k_0$。第 i 轮的 32bits 子密钥 $K_i = k_{31}^i k_{30}^i \cdots k_0^i$ 是由寄存器 K 当前值的最左边 32bits 组成的。因此在第 i 轮，有 $K_i = k_{31}^i k_{30}^i \cdots k_0^i = k_{79} k_{78} \cdots k_{48}$。提取子密钥 K_i 后，对密钥寄存器 $K = k_{79} k_{78} \cdots k_0$ 进行更新，更新步骤如下：

(1) $[k_{79} k_{78} \cdots k_0] = [k_{18} k_{17} \cdots k_{20} k_{19}]$(循环右移 18 位)；

(2) $[k_{79} k_{78} k_{77} k_{76}] = S[k_{79} k_{78} k_{77} k_{76}]$；

(3) $[k_{19} k_{18} k_{17} k_{16} k_{15}] = [k_{19} k_{18} k_{17} k_{16} k_{15}] \oplus rc$。

其中 rc 为轮次数，即 $0 \leqslant rc \leqslant 34$；$S[k]$ 表示用 S 盒进行变换。

4　PFP 算法的效率分析

4.1　PFP 算法的软件效率测试

本节用 C 语言对 PFP 算法和 PRESENT 算法进行编程测试。测试平台：Intel(R)，CPU：Core(TM) i3，主频 2.93 GHz，内存 4GB。两种算法的效率如表 6 所列。

由表 6 可知，PFP 算法的软件加密效率约为 PRESENT 算法的 1.5 倍。

表 6　PFP 和 PRESENT 算法的软件加密效率/Mb/s

算法	PFP	PRESENT
无密钥编排的加密速度	1.5331	0.950
含密钥编排的加密速度	1.4334	0.907

4.2　PFP 算法的硬件效率

PFP 算法的硬件实现所用的触发器主要是用来存储密钥和加密过程中的中间数据。存储 80bits 密钥大约需要 481GE(门电路数)，存储 64bits 数据需要两个 32 位寄存器，大约需要 385GE。轮函数 F 由 3 部分组成，密钥加变换是一个 32 位异或，大概需要 87GE。混淆层有 8 个并行的 4×4 S 盒，大概需要 28.03×8=224.24GE。扩散层 P 可以用电路来实现，所以不需要 GE。每一轮加密过程中最后还需要有左、右两个分支的 32 位异或，这大概需要 87GE。另外，密钥扩展时还有一个 5bits 的异或，大概需要 13.5GE，一个 S 盒大概需要 28.03GE。最后，控制电路和计数电路大概需要 50GE。因此 PFP 算法的硬件实现大约需要 1355GE。表 7 列出了常用轻量级分组密码硬件实现所需的门电路数[9]。

表 7 常用轻量级分组密码硬件实现所需门电路(GE)数

算法	分组长度	密钥长度	GE 数	Logic Process/μm
XTEA	64	128	3490	0.13
HIGHT	64	128	3048	0.25
mCrypton	64	128	2500	0.13
DESXL	64	184	2168	0.18
KATAN	64	80	25.1	0.13
KTANTAN	64	80	25.1	0.13
PRESENT	64	80	1570	0.18
PFP	64	80	1355	0.18

5 PFP 算法的安全性测试与分析

5.1 PFP 算法的依赖性测试

取 10000 个长度为 64bits 的样本进行依赖性测试[10]。PFP 算法的输出长度为 64bits,输入改变 1 位,输出改变的位数 w 期望为输出长度的 1/4～3/4,即 16～48 位,理想值近似等于 1/2。经过测试,对于 34 轮 PFP 算法,输入改变 1 位,输出改变的位数的取值范围为 15～51,平均值接近 32。每一位输出改变的概率 Pr 的取值范围为 0.4834～0.5162,输出改变的概率接近 0.5。测试雪崩效应度 d_a 为0.999150,严格雪崩准则度 d_{sa} 为 0.992024。表 8 给出了 PFP 算法数轮的依赖性测试结果,表 9 给出了 PRESENT 算法数轮的依赖性测试结果。

表 8 PFP 算法的依赖性测试结果

轮数	9	10	11	34
d_a	0.998989	0.999101	0.999049	0.999150
d_{sa}	0.992024	0.992034	0.992173	0.992024
d_c	1.000000	1.000000	1.000000	1.000000
w	15～51	12～50	14～50	15～51
w_a	32.007298	31.998463	32.008244	32.001450
Pr	0.4834～0.5162	0.4826～0.5181	0.4838～0.5210	0.4834～0.5162
Pr_a	0.500114	0.499976	0.500129	0.500114

注:w_a 表示输出改变的平均位数;Pr_a 表示输出改变的平均概率。

表 9 PRESENT 算法的依赖性测试结果

轮数	7	8	9	31
d_a	0.998103	0.998881	0.999166	0.999037
d_{sa}	0.991738	0.992044	0.992029	0.992019
d_c	1.000000	1.000000	1.000000	1.000000
w	13～50	14～50	14～51	14～49
w_a	31.942413	31.992260	32.006454	31.999779
Pr	0.4820～0.5161	0.4785～0.5200	0.4821～0.5257	0.4809～0.5190
Pr_a	0.499100	0.499879	0.500101	0.499997

通过表 8 和表 9 可以得到,PFP 算法从第 10 轮开始充分满足依赖性的要求,PRESENT 算法从第 9 轮开始充分满足依赖性的要求。PRESENT 算法迭代总轮数为 31,所以对 PFP 算法的建议轮数是 34。这说明:AES-128 算法通过 3 轮实现依赖性,迭代总轮数为 10;AES-192 算法通过 3.5 轮实现依赖性,迭代总轮数为 12;AES-256 算法通过 4 轮实现依赖性,迭代总轮数为 14[3-4]。

5.2 PFP 算法的线性分析

对 PFP 算法进行线性分析[11-16]时,利用计算加密过程中线性活动 S 盒的个数来分析算法的线性特征。通过程序计算了 PFP 算法前 20 轮的线性活动 S 盒的个数 LS,计算结果如表 10 所列。通过表 10 可以得到加密 15 轮时 PFP 算法有 56 个活动 S 盒,且 PFP 算法中使用的 S 盒的最大线性偏差为 2^{-2},所以可以得到 PFP 算法的 15 轮最大线性偏差概率为 $DCP_{max}^{15} \leqslant 2^{56-1} \cdot 2^{56 \cdot (-2)} = 2^{-57}$。因此,根据线性复杂度分析,难以找到一个 15 轮的线性壳把加密密文从一个随机置换中区分出来。所以,完整轮数的 PFP 算法足以抵抗线性分析。

表 10 PFP 算法 20 轮线性活动 S 盒的个数

轮数	1	2	3	4	5	6	7	8	9	10
S 盒	0	4	8	12	16	20	24	28	32	36
轮数	11	12	13	14	15	16	17	18	19	20
S 盒	40	44	48	52	56	60	64	68	72	76

5.3 PFP 算法的差分分析

对 PFP 算法进行差分分析[14-20]时,利用计算加密过程中差分活动 S 盒的个数来分析算法的差分特征。这种方法经常用来评估一个算法抵抗差分攻击的性能,而且适用于很多现有的分组密码,如 AES,Camellia,CLEFIA 等。通过程序计算了 PFP 算法前 20 轮的活动 S 盒的个数,计算结果如表 11 所列。

表 11 PFP 算法 20 轮差分活动 S 盒的个数

轮数	1	2	3	4	5	6	7	8	9	10
S 盒	0	3	6	9	13	17	21	25	29	33
轮数	11	12	13	14	15	16	17	18	19	20
S 盒	37	41	45	49	53	57	61	65	69	73

通过表 11 可以得到,加密 15 轮时 PFP 算法有 53 个活动 S 盒,且 PFP 算法中使用的 S 盒的最大差分概率为 2^{-2},所以可以得到 PFP 算法的 15 轮最大差分概率为 $DCP_{max}^{15} \leqslant 2^{53 \cdot (-2)} = 2^{-106}$。虽然 PFP 算法的分组长度只有 64bits,但是上述 15 轮最大差分概率表明 PFP 算法没有明显的 15 轮差分特征。所以,34 轮的 PFP 算法足以抵抗差分分析。

5.4 PFP 算法的不可能差分分析

由于 PFP 算法的整体结构采用 Feistel 结构,轮函数是双射的,因此 PFP 算法存在一个 5 轮不可能差分路径 $\beta_0 \xrightarrow{r-1} \beta_{-1}$。利用此 5 轮不可能差分路径可以计算出第 6 轮密钥的时间复杂度和数据复杂度。

选取 2^{32} 个明文 $P_i=(L_i,R_i)$,$1 \leqslant i \leqslant 2^{32}$,它们的右半部分相同,$C_i=(Z_i,W_i)$ 是对应的密文,计算 $L_i \oplus W_i$,对于 $i \neq j$,寻找 $L_i \oplus W_i = L_j \oplus W_j$。可以得到 2^{31} 个匹配。设 $\alpha = L_i \oplus L_j = W_i \oplus W_j$。

第 6 轮的子密钥可能值有 2^{32} 个,任取一个 K_6,用 K_6 对找到的所有匹配的密文进行解密,如果得到的第 5 轮的差分存在(0α),则此密钥是错误的。对于正确的子密钥 K_6,第 5 轮的差不存在(0α),但对于错误的子密钥值,每个匹配对产生错误密钥值的概率为 2^{-32},因此在对 2^{31} 个匹配实验完后大约有一半子密钥被放弃。反复攻击 32 次,所剩的值中必有正确的子密钥。攻击需要 $32 * 2^{32} = 2^{37}$ 个选择明文,$(2^{32}+2^{31}+\cdots+2+1) * 2^{32} \approx 2^{65}$ 次加密。由于 PFP 算法加密轮数为 34,我们认为完整 PFP 算法针对不可能差分分析[21-23]能够提供足够的安全性。

5.5 PFP 算法的密钥编排攻击

由于密码编排的设计没有一个完整的标准，因此密钥编排设计就有很大的灵活性，而且存在种类繁多的针对密钥编排的特定攻击。针对密钥编排最有效的攻击是相关密钥攻击[24]和滑动攻击，这两种方法都是通过研究不同密钥之间的关系对加密的影响来得到密钥信息。为了抵抗这种攻击，PFP 算法的密钥扩展中用到了一个独立的轮数使得轮密钥不容易产生滑动，而且还用到了非线性操作来混淆密钥寄存器 K 中的内容。尤其是：1)经过 21 轮，密钥寄存器中的所有比特是原始密钥 80bits 的一个非线性函数；2)21 轮后密钥寄存器中的每一个比特至少依赖 4 个原始密钥比特；3)得到的 32 位 K 中，6bit 是关于原始密钥的二次函数，24bit 是关于原始密钥的三次函数，其余比特也是关于原始密钥的六次或者九次函数。因此，这些特征足以抵抗基于密钥编排的攻击。

结束语　本文首先介绍了轻量级分组密码的设计方法，然后设计了一个新的轻量级分组密码 PFP。PFP 算法采用 Feistel 结构，轮函数借鉴了 PRESENT 算法。PFP 算法的分组长度为 64bits，密钥长度为 80bits。分析表明，PFP 算法的硬件实现所需要的门电路数大约为 1355GE，优于 PRESENT 算法，满足资源极端受限环境的需求。测试表明，PFP 的软件加密效率约为 PRESENT 算法的 1.5 倍。通过依赖性测试、线性分析、差分分析、不可能差分分析、密钥编排攻击等方法论证了 PFP 算法的安全性，因此 PFP 算法满足轻量级分组密码的安全需求。

参 考 文 献

[1] 王育民，刘建伟. 通信网的安全——理论与技术[M]. 西安：西安电子科技大学出版社，1999：69-456.

[2] SCHNEIER B. Applied Cryptography: Protocols, algorithms and source code in C[M]. Beijing: China Machine Press, 2000: 1-376.

[3] 冯登国，林东岱，吴文玲. 欧洲信息安全算法工程[M]. 北京：科学出版社，2003：1-190.

[4] 吴文玲，冯登国，张文涛. 分组密码的设计与分析[M]. 北京：清华大学出版社，2009：6-7，217-224，416-420.

[5] AXEL Y P B. Lightweight Cryptography: Cryptographic Engineering for a Pervasive World [D]. Bochum: Ruhr-University Bochum, 2009.

[6] 吴文玲，范伟杰，张蕾. 轻量级分组密码研究进展[M]. 北京：电子工业出版社，2010：140-159.

[7] BOGDANOV A, KNUDSEN L R, LEANDER G, et al. PRESENT: An Ultra-Lightweight Block Cipher[C] // CHES 2007. LNCS 4727, Berlin Heidelberg: Springer-Verlag, 2007: 450-466.

[8] Information technology-Security techniques-Lightweight cryptography-Part 2: Block ciphers: ISO/IEC 29192-2[S]. Geneva: ISO/IEC, 2012.

[9] BADEL S, DAGTEKIN N, JR J N, et al. ARMADILLO: A Multi-purpose Cryptographic Primitive Dedicated to Hardware [M] // Cryptograpic Hardware and Embedded Systems, CHES 2010. LNCS 6225, Berlin Heidelberg: Springer-Verlag, 2010: 398-412.

[10] SERF P. The degrees of completeness, of avalanche effect, and of strict avalanche criterion for mars, rc6, rijndael, serpent, and twofish with reduced number of rounds[EB/OL]. http://www.cosic.esat.kuleuven.ac.be/nessie/reports/phase1/sagwp3-003.pdf, 2000-2-3.

[11] MATSUI M. Linear Cryptanalysis Method for DES Cipher [C] // Advances in Cryptology-EUROCRYPT'93, LNCS 765. Berlin Heidelberg: Springer-Verlag, 1993: 386-397.

[12] 冯登国. 密码分析学[M]. 北京：清华大学出版社，2000：58-59.

[13] SHI Y Y. Research and Design of Block Cipher Algorithms [D]. Nanjing: Nanjing University of Aeronautics & Astronautics, 2014. (in Chinese)
时阳阳. 分组密码算法的研究与设计[D]. 南京：南京航空航天大学，2014.

[14] KANDA M, TAKASHIMA Y, MATSUMOTO T, et al. A Strategy for Constructing Fast Round Functions with Practical Security Against Differential and Linear Cryptanalysis[C] // SAC'98. LNCS 1556, Berlin Heidelberg: Springer-Verlag, 1999: 264-279.

[15] KANDA M. Practical Security Evaluation against Differential and Linear Cryptanalysis for Feistel Ciphers with SPN Round Function [C] // SAC 2000. LNCS 2012, Berlin Heidelberg: Springer-Verlag, 2012: 324-338.

[16] HONG S, LEE S, LIM J, et al. Provable Security against Differential and Linear Cryptanalysis for the SPN Structure[C] // FSE 2000. LNCS 1978, Berlin Heidelberg: Springer-Verlag, 2001: 273-283.

[17] BIHAN E, SHAMIR A. Differential cryptanalysis of the data encryption standard[M]. New York: Springer-Verlag, 1993.

[18] BIHAN E, SHAMIR A. Differential Cryptanalysis of DES-like Cryptosystems[J]. Journal of Cryptology, 1991, 4(1): 3-72.

[19] LAI X, MASSEY J L. Markov Ciphers and Differential Cryptanalysis[M] // Advances in Cryptology-ENCRYPT'91. LNCS 547, Berlin Heidelberg: Springer-Verlag, 1991: 17-38.

[20] NYBERG K, KNUDSEN L R. Provable Security Against a Differential Attack[J]. Journal of Cryptology, 1995, 8(1): 27-37.

[21] BIHAM E, BIRYUKOV A, SHAMIR A. Cryptanalysis of Skipjack Reduced to 31 Rounds using Impossible Differentials[M] // Advances in Cryptology-EUROCRYPT'99. LNCS 3027, Berlin Heidelberg: Springer-Verlag, 1999: 12-23.

[22] BIHAM E. Cryptanalysis of Skipjack Reduced to 31 Rounds Using Impossible Differential[J]. Journal of Cryptology, 2005, 18(4): 291-311.

[23] KIM J, HONG S, SUNG J, et al. Impossible Differential Cryptanalysis for Block Cipher Structures[C] // INDOCRYPT 2003. LNCS 2904, Berlin Heidelberg: Springer-Verlag, 2003: 82-96.

[24] BIHAM E. New types of cryptanalytic attacks using related keys[J]. Journal of Cryptology, 1994, 7(4): 229-246.