密码学报 ISSN 2095-7025 CN 10-1195/TN
Journal of Cryptologic Research, 2019, 6(6): 690–703

E-mail: jcr@cacrnet.org.cn
http://www.jcr.cacrnet.org.cn
Tel/Fax: +86-10-82789618

分组密码专刊

分组密码 uBlock*

吴文玲[1,2], 张 蕾[1], 郑雅菲[1], 李灵琛[1,2]

1. 中国科学院 软件研究所 可信计算与信息保障实验室, 北京 100190
2. 中国科学院大学, 北京 100049
通信作者: 吴文玲, E-mail: wwl@tca.iscas.ac.cn

摘 要: 本文首先介绍分组密码 uBlock 算法, 然后简要介绍 uBlock 的设计原理, 初步的安全性分析评估, 以及各种平台的实现性能等. uBlock 是一族分组密码算法, 分组长度和密钥长度支持 128 和 256 比特. uBlock 算法的整体结构、S 盒、扩散矩阵、密钥扩展等设计, 处处体现了安全、实现效率以及适应性的平衡. uBlock 算法对差分分析、线性分析、积分分析、不可能差分分析、中间相遇攻击等分组密码分析方法具有足够的安全冗余. uBlock 算法适应各种软硬件平台; 充分考虑了现代微处理器的计算资源, 可以利用 SSE 和 AVX2 等指令集高效实现; 硬件实现简单而有效, 既可以高速实现, 保障高性能环境的安全应用, 也可以轻量化实现, 满足资源受限环境的安全需求.

关键词: 分组密码; PX 结构; S 盒; 扩散; 安全性分析; 指令集
中图分类号: TP309.7 **文献标识码:** A DOI: 10.13868/j.cnki.jcr.000334

中文引用格式: 吴文玲, 张蕾, 郑雅菲, 李灵琛. 分组密码 uBlock[J]. 密码学报, 2019, 6(6): 690–703.
英文引用格式: WU W L, ZHANG L, ZHENG Y F, LI L C. The block cipher uBlock[J]. Journal of Cryptologic Research, 2019, 6(6): 690–703.

The Block Cipher uBlock

WU Wen-Ling[1,2], ZHANG Lei[1], ZHENG Ya-Fei[1], LI Ling-Chen[1,2]

1. Trusted Computing and Information Assurance Laboratory, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
2. University of Chinese Academy of Sciences, Beijing 100049, China
Corresponding author: WU Wen-Ling, E-mail: wwl@tca.iscas.ac.cn

Abstract: This paper first presents a detailed introduction of block cipher uBlock, then gives a brief description of the design principle, preliminary security evaluation, and implementation performance on various platforms, etc. uBlock is a family of block ciphers supporting 128-bit and 256-bit block sizes and key sizes. There are different versions of uBlock, which are denoted as uBlock-128/128, uBlock-128/256, and uBlock-256/256. The balance between security, implementation performance, and adaptability is reflected in the overall algorithm design, S-box, diffusion matrix, key schedule, and

* 基金项目: 国家自然科学基金项目 (61672509); 国家密码发展基金 (MMJJ20170101)
Foundation: National Natural Science Foundation of China (61672509); National Cryptography Development Fund (MMJJ20170101)
收稿日期: 2019-11-15 定稿日期: 2019-12-10

other details. uBlock has sufficient security redundancy against many cryptanalyses of block ciphers, such as differential cryptanalysis, linear cryptanalysis, integral cryptanalysis, impossible differential cryptanalysis, and meet in the middle attack. uBlock fully considers the computing resources of modern microprocessors, and can be efficiently implemented with SSE and AVX2 instruction sets. The hardware implementation of uBlock is simple and efficient. It can be implemented at high speed to ensure the security for high-performance environments, and it can also be implemented in lightweight manner to satisfy the security requirement of resource constrained environments.

Key words: block cipher; PX structure; S-box; diffusion; cryptanalysis; instruction set

1 uBlock 分组密码

uBlock 是一族分组密码算法, 分组长度和密钥长度支持 128 和 256 比特, 记为 uBlock-128/128、uBlock-128/256 和 uBlock-256/256, 它们的迭代轮数 r 分别为 16、24 和 24.

1.1 符号

本文使用了如下符号.

X:	n 比特明文	Y:	n 比特密文
K:	k 比特密钥	RK^i:	n 比特轮密钥
$\mathrm{PL}_n, \mathrm{PR}_n, \mathrm{PL}_n^{-1}, \mathrm{PR}_n^{-1}$:	$n/16$ 个字节的向量置换	s:	4 比特 S 盒
$S_n(S_n^{-1})$	$n/8$ 个 s 盒 (逆) 的并置	S_k:	$k/16$ 个 s 盒的并置
PK_1:	16 个半字节的向量置换	$\mathrm{PK}_2, \mathrm{PK}_3$:	32 个半字节的向量置换
$\oplus$:	模 2 加运算	$\lll b$:	循环左移 b 比特
$\lll_{32} b$:	分块 32 比特循环左移 b 比特	$\|$:	比特串的连接

1.2 加密算法

加密算法由 r 轮迭代变换组成, 轮变换如图 1 所示.

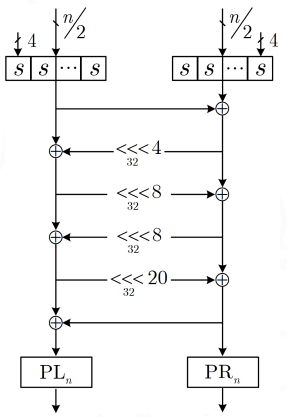

图 1 加密轮变换

Figure 1 Round encryption

输入 n 比特明文 X 和轮密钥 $\mathrm{RK}^0, \mathrm{RK}^1, \cdots, \mathrm{RK}^r$, 输出 n 比特密文 Y. 加密算法 E 如下:

$$\begin{aligned}
&\mathrm{uBlock.Enc}\left(X, \mathrm{RK}^0, \mathrm{RK}^1, \cdots, \mathrm{RK}^r\right)\\
&X_0 \| X_1 \leftarrow X\\
&\text{for } i = 0 \text{ to } r-1 \text{ do}
\end{aligned}$$

$$
\begin{aligned}
&\mathrm{RK}_0^i \| \mathrm{RK}_1^i \leftarrow \mathrm{RK}^i \\
&X_0 \leftarrow S_n\left(X_0 \oplus \mathrm{RK}_0^i\right) \\
&X_1 \leftarrow S_n\left(X_1 \oplus \mathrm{RK}_1^i\right) \\
&X_1 \leftarrow X_1 \oplus X_0 \\
&X_0 \leftarrow X_0 \oplus \left(X_1 \lll_{32} 4\right) \\
&X_1 \leftarrow X_1 \oplus \left(X_0 \lll_{32} 8\right) \\
&X_0 \leftarrow X_0 \oplus \left(X_1 \lll_{32} 8\right) \\
&X_1 \leftarrow X_1 \oplus \left(X_0 \lll_{32} 20\right) \\
&X_0 \leftarrow X_0 \oplus X_1 \\
&X_0 \leftarrow \mathrm{PL}_n\left(X_0\right) \\
&X_1 \leftarrow \mathrm{PR}_n\left(X_1\right) \\
&Y \leftarrow \mathrm{RK}^r \oplus \left(X_0 \| X_1\right)
\end{aligned}
$$

其中基本模块定义如下:

(1) $\boldsymbol{S_n}$. S_n 由 $n/8$ 个相同的 4 比特 S 盒并置而成, 定义如下:

$$
\begin{aligned}
&S_n : \left(\{0,1\}^4\right)^{n/8} \rightarrow \left(\{0,1\}^4\right)^{n/8} \\
&\quad \left(x_0, x_1, \cdots, x_{n/8-1}\right) \rightarrow \left(s\left(x_0\right), s\left(x_1\right), \cdots, s\left(x_{n/8-1}\right)\right)
\end{aligned}
$$

4 比特 S 盒如表 1 所示.

表 **1** 4 比特 S 盒 ($\boldsymbol{s}$)
Table 1 4-bit S-box ($\boldsymbol{s}$)

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\boldsymbol{s(x)}$	7	4	9	c	b	a	d	8	f	e	1	6	0	3	2	5

(2) $\mathbf{PL_n}$ 和 $\mathbf{PR_n}$. PL_n 和 PR_n 都是 $m(=n/16)$ 个字节的向量置换, 具体见表 2.

表 **2** $\mathrm{PL}_{\boldsymbol{n}}$ 和 $\mathrm{PR}_{\boldsymbol{n}}$
Table 2 $\mathrm{PL}_{\boldsymbol{n}}$ and $\mathrm{PR}_{\boldsymbol{n}}$

PL_{128}	{1,3,4,6,0,2,7,5}
PR_{128}	{2,7,5,0,1,6,4,3}
PL_{256}	{2,7,8,13,3,6,9,12,1,4,15,10,14,11,5,0}
PR_{256}	{6,11,1,12,9,4,2,15,7,0,13,10,14,3,8,5}

比如 PL_{128} 的表达式为:

$$
\begin{aligned}
&\mathrm{PL}_{128} : \left(\{0,1\}^8\right)^8 \rightarrow \left(\{0,1\}^8\right)^8 \\
&\quad \left(y_0, y_1, y_2, \cdots, y_6, y_7\right) \rightarrow \left(z_0, z_1, z_2, \cdots, z_6, z_7\right) \\
&\quad z_0 = y_1, \quad z_1 = y_3, \quad z_2 = y_4, \quad z_3 = y_6, \\
&\quad z_4 = y_0, \quad z_5 = y_2, \quad z_6 = y_7, \quad z_7 = y_5
\end{aligned}
$$

1.3 解密算法

解密算法由 r 轮迭代变换组成, 输入 n 比特密文 Y, 轮密钥 $\mathrm{RK}^r, \mathrm{RK}^{r-1}, \cdots, \mathrm{RK}^0$, 输出 n 比特明文 X. 解密算法如下:

$$
\begin{aligned}
&\mathrm{uBlock.Dec}\left(Y, \mathrm{RK}^r, \mathrm{RK}^{r-1}, \cdots, \mathrm{RK}^0\right)\\
&Y_0 \| Y_1 \leftarrow Y\\
&\text{for } i = r \text{ to } 1 \text{ do}\\
&\quad \mathrm{RK}_0^i \| \mathrm{RK}_1^i \leftarrow \mathrm{RK}^i\\
&\quad Y_0 \leftarrow Y_0 \oplus \mathrm{RK}_0^i\\
&\quad Y_1 \leftarrow Y_1 \oplus \mathrm{RK}_1^i\\
&\quad Y_0 \leftarrow \mathrm{PL}_n^{-1}\left(Y_0\right)\\
&\quad Y_1 \leftarrow \mathrm{PR}_n^{-1}\left(Y_1\right)\\
&\quad Y_0 \leftarrow Y_0 \oplus Y_1\\
&\quad Y_1 \leftarrow Y_1 \oplus \left(Y_0 \underset{32}{\lll} 20\right)\\
&\quad Y_0 \leftarrow Y_0 \oplus \left(Y_1 \underset{32}{\lll} 8\right)\\
&\quad Y_1 \leftarrow Y_1 \oplus \left(Y_0 \underset{32}{\lll} 8\right)\\
&\quad Y_0 \leftarrow Y_0 \oplus \left(Y_1 \underset{32}{\lll} 4\right)\\
&\quad Y_1 \leftarrow Y_1 \oplus Y_0\\
&\quad Y_0 \leftarrow S_n^{-1}\left(Y_0\right)\\
&\quad Y_1 \leftarrow S_n^{-1}\left(Y_1\right)\\
&X \leftarrow \mathrm{RK}^0 \oplus \left(Y_0 \| Y_1\right)
\end{aligned}
$$

其中 S_n^{-1}、PL_n^{-1} 和 PR_n^{-1} 分别是 S_n、PL_n 和 PR_n 的逆, 具体见表 3 和表 4.

表 **3** 4 比特 S 盒的逆 ($\boldsymbol{s^{-1}}$)
Table 3 Inverse of 4-bit S-box ($\boldsymbol{s^{-1}}$)

$\boldsymbol{x}$	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
$\boldsymbol{s^{-1}(x)}$	c	a	e	D	1	f	B	0	7	2	5	4	3	6	9	8

表 **4** $\mathrm{PL}_{\boldsymbol{n}}^{-\mathbf{1}}$ 和 $\mathrm{PR}_{\boldsymbol{n}}^{-\mathbf{1}}$
Table 4 $\mathrm{PL}_{\boldsymbol{n}}^{-\mathbf{1}}$ and $\mathrm{PR}_{\boldsymbol{n}}^{-\mathbf{1}}$

PL_{128}^{-1}	{4,0,5,1,2,7,3,6}
PR_{128}^{-1}	{3,4,0,7,6,2,5,1}
PL_{256}^{-1}	{15,8,0,4,9,14,5,1,2,6,11,13,7,3,12,10}
PR_{256}^{-1}	{9,2,6,13,5,15,0,8,14,4,11,1,3,10,12,7}

1.4 密钥扩展算法

将 k 比特密钥 K 放置在 k 比特寄存器, 取寄存器的左 n 比特作为轮密钥 RK^0, 然后对 $i = 1, 2, \cdots, r$, 更新寄存器, 并取寄存器的左 n 比特作为轮密钥 RK^i.

寄存器更新方式如下 (如图2所示):

$$
\begin{aligned}
&K_0\|K_1\|K_2\|K_3 \leftarrow K\\
&K_0\|K_1 \leftarrow \mathrm{PK}_t\,(K_0\|K_1)\\
&K_2 \leftarrow K_2 \oplus S_k\,(K_0 \oplus \mathrm{RC}_i)\\
&K_3 \leftarrow K_3 \oplus T_k\,(K_1)\\
&K \leftarrow K_2\|K_3\|K_1\|K_0
\end{aligned}
$$

其中 S_k 是 $k/16$ 个 4 比特 S 盒的并置, T_k 是对 K_1 的每半字节 $\otimes 2$, 有限域 $\mathrm{GF}(2^4)$ 的不可约多项式 $m(x)=x^4+x+1$; RC_i 为 32 比特常数, 作用在 K_0 的左 32 比特. PK_t 有三种情况, $t=1,2,3$, PK_1、PK_2 和 PK_3 分别用于 uBlock-128/128、uBlock-128/256 和 uBlock-256/256 的密钥扩展算法. PK_1 是 16 个半字节的向量置换, PK_2 和 PK_3 都是 32 个半字节的向量置换, 具体见表5.

表 **5** PK_t
Table 5 PK_t

PK_1	{6,0,8,13,1,15,5,10,4,9,12,2,11,3,7,14}
PK_2	{10,5,15,0,2,7,8,13,14,6,4,12,1,3,11,9,24,25,26,27,28,29,30,31,16,17,18,19,20,21,22,23}
PK_3	{10,5,15,0,2,7,8,13,1,14,4,12,9,11,3,6,24,25,26,27,28,29,30,31,16,17,18,19,20,21,22,23}

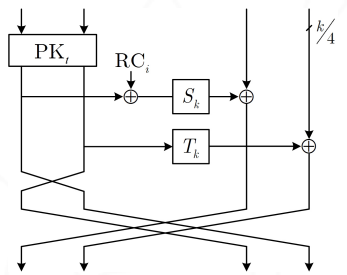

图 2 密钥状态更新函数
Figure 2 Update function of key state

32 比特常数 RC_i 由 8 级 LFSR 生成, 初始条件为 $c_0=c_3=c_6=c_7=0, c_1=c_2=c_4=c_5=1$; 对 $i\geq 8$, $c_i=c_{i-2}\oplus c_{i-3}\oplus c_{i-7}\oplus c_{i-8}$. 令

$$
\begin{aligned}
a_i &= c_i\bar{c}_{i+1}c_{i+2}c_{i+3}c_{i+4}c_{i+5}c_{i+6}c_{i+7}\\
a_i' &= c_i\bar{c}_{i+1}c_{i+2}\bar{c}_{i+3}c_{i+4}\bar{c}_{i+5}c_{i+6}c_{i+7}\\
a_i'' &= c_ic_{i+1}c_{i+2}\bar{c}_{i+3}c_{i+4}c_{i+5}c_{i+6}\bar{c}_{i+7}\\
a_i''' &= c_ic_{i+1}c_{i+2}c_{i+3}c_{i+4}\bar{c}_{i+5}c_{i+6}\bar{c}_{i+7}
\end{aligned}
$$

则 $\mathrm{RC}_i = a_i\|a_i'\|a_i''\|a_i'''$.

常数 RC_i $(i=1,2,\cdots,24)$ 的 16 进制表示如表6.

2 uBlock 的设计原理

2.1 整体结构

uBlock 的整体结构称为 PX 结构, 如图3所示. PX 结构是 SP 结构的一种细化结构, PX 是 Pshufb-Xor 的缩写, Pshufb 和 Xor 分别是向量置换和异或运算指令. 采用 S 盒和分支数的理念, PX 结构针

表 **6** 常数 RC_i
Table 6 Constants RC_i

轮常数	值	轮常数	值	轮常数	值
RC_1	988cc9dd	RC_9	392d687c	RC_{17}	8296d3c7
RC_2	f0e4a1b5	RC_{10}	b3a7e2f6	RC_{18}	c5d19480
RC_3	21357064	RC_{11}	a7b3f6e2	RC_{19}	4a5e1b0f
RC_4	8397d2c6	RC_{12}	8e9adfcb	RC_{20}	55410410
RC_5	c7d39682	RC_{13}	dcc88d99	RC_{21}	6b7f3a2e
RC_6	4f5b1e0a	RC_{14}	786c293d	RC_{22}	17034652
RC_7	5e4a0f1b	RC_{15}	30246175	RC_{23}	effbbeaa
RC_8	7c682d39	RC_{16}	a1b5f0e4	RC_{24}	1f0b4e5a

对差分分析和线性分析具有可证明的安全性, 对于不可能差分分析、积分分析、中间相遇攻击等分析方法具有相对成熟的分析评估理论支持. 在同等安全情况下, PX 结构具有更好的软件和硬件实现性能. 利用 SSE/AVX 指令集提供的 128/256 比特寄存器的异或运算和向量置换指令, 对于由 4 比特 S 盒构造的非线性变换层, 以及线性变换中基于 4 比特的向量置换均仅需一条指令即可实现; 因此, 该加密方法的软件实现每轮变换仅需要 m 条异或指令和 $m+4$ 条向量置换指令. 此外, 该实现方法不需要查表操作, 不仅可以提供高性能软件实现, 还可抵抗缓存计时等侧信道攻击. 该结构采用 4 比特 S 盒构造非线性变换层, 相较于 AES 等算法采用的 8 比特 S 盒, 更易于硬件实现, 延迟、面积、能耗等指标更优. 其次, 4 比特 S 盒在各种平台的软件实现灵活, 可以采用查表方式, 也可以采用比特切片的实现方式. 此外, 该结构非常灵活, 根据 $P_j(j=1,\cdots,m)$、P_{m+1} 和 P_{m+2} 等不同的参数选择都有相应的轮函数与之对应, 便于设计多参数规模、安全性高且实现代价低的分组密码算法族. $P_j(j=1,\cdots,m)$ 可以选择面向字的置换, 也可以选择分块循环移位, 设计者可以灵活选取分块大小 a 和移位数 $b_j(j=1,\cdots,m)$.

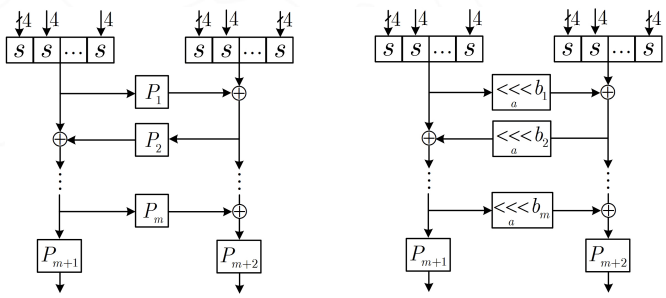

图 3 PX 整体结构
Figure 3 PX framework

2.2 S 盒

uBlock 算法采用 4 比特 S 盒, 一个原因是 4 比特 S 盒可以用 SSE 等指令集快速实现, 另一个原因是硬件实现代价的考虑. 和 8 比特 S 盒相比, 4 比特 S 盒具有明显的硬件优势, 不仅硬件实现代价小, 而且延迟能耗等方面都有明显优势.

uBlock 算法的 S 盒如图 4 所示, 需要 2 个与非门、2 个或非门、2 个异或门、2 个异或非门, 关键路径为 4, 每比特平均 1 个乘法电路. 针对侧信道攻击的防护, 属于最易门限实现的类别. 在安全性方面, uBlock 算法的 S 盒没有不动点、最大线性概率达到最佳 2^{-2}、最大差分概率达到最佳 2^{-2}、代数次数达到最佳 3. 最后, uBlock 算法中加密和密钥扩展使用同一个 4 比特 S 盒, 可以实现最大程度的硬件电路模块复用, 有效降低硬件实现规模和代价.

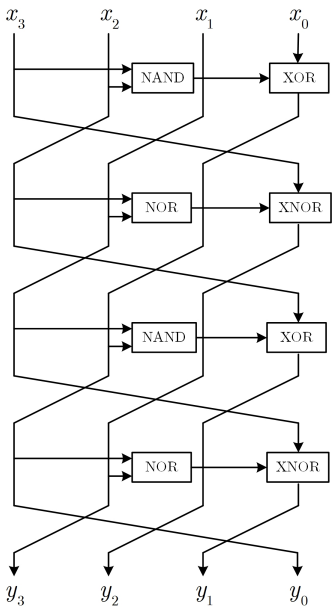

图 4 S 盒实现流程

Figure 4 Implementation process of S-box

2.3 扩散层

uBlock 算法的扩散层由两部分组成, 一个是 PX 结构 (图 3) 中的 $P_j(j=1,\cdots,m)$ 及异或运算, 记为线性变换 B, 另一个是 PX 结构中的 P_{m+1} 和 P_{m+2}. 考虑到算法的软件实现性能和适应性, $P_j(j=1,\cdots,m)$ 选择为分块 32 的循环移位, 而且期望 m 尽可能小, b_j 尽可能为 0 或者是 8 的倍数. m 的选取和分块大小 a 有关, 也和扩散层的分支数有关. 分块大小为 32, 采用 4 比特 S 盒, 利用 Feistel 结构构造 16×16 的二元域最优扩散层, 最少需迭代 6 次, 即 $m=6$. 依据我们关于二元域最优扩散层的研究成果 [1], 选取 $b_1=b_6=0$, $b_2=4$, $b_3=b_4=8$, $b_5=20$, 保证 16×16 的二元域矩阵分支数为 8, 达到二元域上的最佳.

P_7 和 P_8 对应算法中的 PL_n 和 PR_n, 它们都是面向字节的向量置换, 具有最佳的硬件性能, 而且适宜 8 位处理器. 线性变换 B 和 PL_n 以及 PR_n 的联合使用, 使得 uBlock 算法的整体结构对于差分分析和线性分析等分析方法具有可证明安全性. 经过反复分析测试, 我们选定目前采用的 PL_n 和 PR_n, uBlock-128 和 uBlock-256 的全扩散轮数分别为 2 和 3, 差分和线性活动 S 盒的个数、不可能差分和积分区分器等达到设计要求.

2.4 密钥扩展算法

密钥扩展算法采用随用即生成的方式生成轮密钥, 不增加存贮负担; 状态更新函数简单且可逆, 使得密钥扩展算法的实现更加灵活. 密钥扩展算法采用与加密算法不同的整体结构, 利用向量置换构造了扩展 Feistel 结构, 相比广义 Feistel 结构扩散更快, 相比 Feistel 结构更轻量. 安全性方面主要考虑的是双系攻击和相关密钥类攻击的抵抗力, 密钥扩展算法的扩散性, 轮密钥和种子密钥的关系. 实现性能方面, 充分利用向量置换, 尽量不增加实现成本, 为资源受限环境的应用提供保障. 向量置换的设计主要考虑密钥扩展算法的扩散性和相关密钥活跃 S 盒的个数等因素, 其次兼顾硬件和软件实现性能. S 盒的个数是安全性和实现性能的折中结果. $\otimes2$ 操作以很小的代价提供半字节内部的扩散, 硬件实现 $\otimes2$ 操作仅需 1 比特异或. 同时, 向量置换、非线性 S 盒与状态字间的异或操作的联合使用, 提供了足够的非线性和良好的扩散效果, 能够使得算法抵抗双系攻击等分析方法.

3 uBlock 的安全性

本部分简要介绍 uBlock 算法针对典型分组密码分析方法的安全性评估结果. 为了方便, uBlock-256/256 简写为 uBlock-256, uBlock-128 指分组长度为 128 比特的 uBlock.

3.1 差分分析

针对差分分析 [2], 采用搜索差分活跃 S 盒个数的方法评估算法抵抗攻击的能力. 该方法也被其它大量算法的安全性评估所采用, 如 Camellia [3]、CLEFIA [4]、LBlock [5] 等. 我们通过计算机搜索了 uBlock-128 算法的差分活跃 S 盒个数, 结果见表 7. 10 轮 uBlock-128 算法至少有 66 个差分活跃 S 盒, 由于算法采用的 S 盒的最大差分概率为 2^{-2}, 因此 10 轮 uBlock-128 算法的最大差分路径概率满足 $\mathrm{DCP}_{\max}^{10r} \leq 2^{66\times(-2)} = 2^{-132}$, 说明 10 轮 uBlock-128 已经不存在差分分析可利用的有效差分路径. 考虑到 uBlock-128 的迭代轮数和全扩散轮数, 可以相信全轮 uBlock-128 算法针对差分分析是安全的.

表 7 uBlock-128 的活跃 S 盒数

Table 7 Active S-boxes of uBlock-128

轮数	1	2	3	4	5	6	7	8	9	10
活跃 S 盒数	1	8	13	24	30	36	43	50	56	66

对于 uBlock-256, 利用超级 S 盒和分支数, 可以证明 4 轮 uBlock-256 至少有 32 个差分活跃 S 盒. 因此, 16 轮 uBlock-256 至少有 128 个差分活跃 S 盒, 说明 uBlock-256 最长存在 15 轮的有效差分路径. 进一步, 考虑到 uBlock-256 的全扩散轮数为 3, 迭代轮数为 24, 因此, 全轮 uBlock-256 算法针对差分分析是安全的.

3.2 线性分析

针对线性分析 [6], 我们也采用类似的搜索活跃 S 盒个数的方法来评估算法抵抗攻击的能力. 对于 uBlock-128 算法, 线性活跃 S 盒个数的测试结果同上小节的差分测试结果. 对于 uBlock-256 算法, 可以证明 4 轮至少有 32 个线性活跃 S 盒. 由于 uBlock 算法使用的 S 盒的最大线性概率为 2^{-2}, 因此, 考虑到 uBlock 算法的迭代轮数和全扩散轮数, 可以相信 uBlock 算法针对线性分析是安全的.

3.3 不可能差分分析

对基于半字节的 uBlock 算法, 以半字节为单位引入变量, 即所有变量都取值于有限域 GF(2^4), 选择输入和输出差分的汉明重量为 1, 搜索最长轮数的不可能差分.

对于 uBlock-128 算法, 构造了 1024 条 4 轮不可能差分, 具体如表 8.

表 8 uBlock-128 的 4 轮不可能差分

Table 8 4-round impossible differentials of uBlock-128

序号	输入差分	输出差分
1	10000000000000000000000000000000	10000000000000000000000000000000
2	10000000000000000000000000000000	01000000000000000000000000000000
3	10000000000000000000000000000000	00100000000000000000000000000000
...	...	...
1023	00000000000000000000000000000001	00000000000000000000000000000010
1024	00000000000000000000000000000001	00000000000000000000000000000001

对于 uBlock-256 算法, 构造了 4096 条 5 轮不可能差分, 具体如表 9.

利用上述不可能差分区分器, 可以给出缩减轮 uBlock 的不可能差分分析. 考虑到 uBlock 算法的扩散性和迭代轮数, 可以相信全轮 uBlock 针对不可能差分分析提供了足够的安全冗余.

3.4 积分分析

uBlock 算法整体结构为 SP, S 盒的代数次数为 3, 因此, uBlock-128 是 (4,3,32)-SPN, uBlock-256 是 (4,3,64)-SPN. 依据 Todo 的搜索结果可知, uBlock-128 存在 7 轮积分区分器, 数据复杂度为 2^{124} 选择明文; uBlock-256 存在 8 轮积分区分器, 数据复杂度为 2^{252} 选择明文. 具体如下.

表 **9** uBlock-256 的 5 轮不可能差分
Table 9 5-round impossible differentials of uBlock-256

序号	输入差分	输出差分
1	10000000000000000000000000000000 00000000000000000000000000000000	10000000000000000000000000000000 00000000000000000000000000000000
2	10000000000000000000000000000000 00000000000000000000000000000000	01000000000000000000000000000000 00000000000000000000000000000000
3	10000000000000000000000000000000 00000000000000000000000000000000	00100000000000000000000000000000 00000000000000000000000000000000
...	...	...
4095	00000000000000000000000000000000 00000000000000000000000000000001	00000000000000000000000000000000 00000000000000000000000000000010
4096	00000000000000000000000000000000 00000000000000000000000000000001	00000000000000000000000000000000 00000000000000000000000000000001

对于 uBlock-128 算法, 任意选取一个半字节取常数, 其余 31 个半字节活跃, 7 轮 uBlock-128 加密之后, 2^{124} 个密文的异或和为全 0, 如表 10 所示.

表 **10** uBlock-128 的 7 轮积分区分器
Table 10 7-round integral distinguishers of uBlock-128

序号	输入	输出异或和
1	caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	00000000000000000000000000000000
2	acaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	00000000000000000000000000000000
...	...	...
32	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac	00000000000000000000000000000000

其中 c 表示常数, a 表示活跃.

对于 uBlock-256 算法, 任意选取一个半字节取常数, 其余 63 个半字节活跃, 8 轮 uBlock-256 加密之后, 2^{252} 个密文的异或和为全 0, 如表 11 所示.

表 **11** uBlock-256 的 8 轮积分区分器
Table 11 8-round integral distinguishers of uBlock-256

序号	输入	输出异或和
1	caaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	00000000000000000000000000000000 00000000000000000000000000000000
2	acaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	00000000000000000000000000000000 00000000000000000000000000000000
...	...	...
64	aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac	00000000000000000000000000000000 00000000000000000000000000000000

文献 [7] 针对可分性传播过程中计算量快速增大的问题, 提出了一种提前删除冗余向量的技术. 文献 [8] 将可分性扩展到比特可分性, 对可分性的刻画更准确. 文献 [9] 基于 MILP 给出了更有效的积分区分器搜索算法. 利用这些新技术有可能给出 uBlock 更好的积分区分器; 但是考虑到 uBlock 算法的扩散性和迭代轮数, 可以相信 uBlock 针对积分分析提供了足够的安全冗余.

3.5 中间相遇攻击

我们评估了 uBlock 算法抵抗 δ-集中间相遇攻击的能力, 结果显示 6 轮 uBlock-128/128、8 轮 uBlock-128/256、6 轮 uBlock-256/256 存在此类攻击, 攻击过程中的主要参数以及攻击结果汇总如表 12.

此外, 我们评估了 uBlock 算法抵抗双系攻击的能力, 结果显示 11 轮 uBlock-128/128、16 轮 uBlock-128/256 以及 13 轮 uBlock-256/256 存在双系攻击; 全轮 uBlock 算法不存在有效的双系攻击.

表 **12** uBlock 算法的中间相遇攻击结果
Table 12 Results of uBlock against meet in the middle attack

算法名称	攻击轮数	区分器长度	数据	时间	存储
uBlock-128/128	6	3	2^{60}	$2^{125.42}$	$2^{103.9}$
uBlock-128/256	8	4	2^{64}	2^{245}	$2^{238.99}$
uBlock-256/256	6	3	2^{64}	$2^{117.42}$	$2^{89.99}$

4 uBlock 的实现性能

uBlock 算法的主要运算仅包含 4 比特 S 盒、以 4 比特/8 比特为单位的置换、以及异或等逻辑运算, 非常适合现代 CPU 支持的 SIMD 指令集实现. 并行结构、简单的 S 盒和线性变换, 使得 uBlock 算法的硬件实现简单而有效, 既可以高速实现, 也可以轻量化实现.

4.1 PC 软件性能

对于 uBlock 算法在 32/64 位平台上的实现性能, 可以采用传统的查表方式实现, 也可以利用现代 CPU 支持的 SIMD 指令集实现, 比如 Intel CPU 支持的 SSE 等指令集, 可以支持 128 比特寄存器和相应的逻辑运算、向量置换等. 考虑到 uBlock 算法的主要运算仅包含 4 比特 S 盒、以 4 比特/8 比特为单位的置换、以及异或等逻辑运算, 非常适合用向量置换指令.

我们实际测试了 uBlock 算法在软件平台上的实现速度. 测试环境为: CPU Intel Core i7-3740QM CPU @ 2.70 GHz, 内存 16.0 GB, 操作系统 Windows 7 专业版 64 位, 编译环境 Microsoft Visual Studio 2010. 随机生成明文和密钥, 测试运行 100 次, 取平均速度, 速度单位 Mbps. uBlock 算法各版本针对不同长度的消息 (包括短消息和长消息) 的软件实现速度检测结果如表 13 所示.

表 **13** uBlock 算法的软件实现性能
Table 13 Software implementation performance of uBlock

消息长度	算法版本	加密速度		解密速度	
		Cycle/Byte	**Mbps**	**Cycle/Byte**	**Mbps**
256 Bytes	uBlock-128/128	13.1	1641	13.0	1670
	uBlock-128/256	21.1	1056	20.1	1098
	uBlock-256/256	17.1	1268	16.8	1276
1 MB	uBlock-128/128	11.8	1809	11.3	1869
	uBlock-128/256	17.4	1216	16.7	1289
	uBlock-256/256	13.6	1566	13.4	1571

根据测试结果, 在长消息下采用 SSE 指令集实现的 uBlock-128/128 算法的加密速度可以达到 11.8 Cycles/Byte (1809 Mbps), 解密速度可以达到 11.3 Cycles/Byte (1869 Mbps); uBlock-256/256 算法的加密速度可以达到 13.6 Cycles / Byte (1566 Mbps), 解密速度可以达到 13.4 Cycles/Byte (1571 Mbps). 进一步, 如果采用最新版本 CPU 支持的 AVX2 指令, uBlock-256/256 算法的实现速度有提升, 而且随着 AVX 指令集的更新以及寄存器长度的扩展, uBlock-256/256 算法的性能优势将会展现.

4.2 ARM 软件性能

对于 uBlock 算法在 32 位 ARM 平台上的实现性能, 与 64 位 Windows 平台类似, 大部分 ARM 平台处理器也提供了扩展指令集——neon 指令集, 支持相应的 SIMD 运算, 包括向量置换、128 比特逻辑运算等. 基于此可以给出 uBlock 算法在 ARM 平台下基于 neon 指令集的优化实现. 此外, 对于嵌入式微处理器中使用的低功耗 ARM, 计算能力和 RAM/ROM 等资源可能受限, 可以直接按照算法描述给出 uBlock 算法的基础实现或查大表实现.

我们实际测试了 uBlock 算法在 32 位 ARM 软件平台上的加/解密速度, 如表 14 所示. 测试环境为 ARM Cortex A53, 主频 1.4 GHz. 随机生成明文和密钥, 测试运行 100 次, 取平均速度, 速度单位 Mbps.

表 **14** uBlock 算法的 ARM 平台软件实现性能

Table 14 Software implementation performance of ARM platform of uBlock

消息长度	算法版本	加密速度 (Mbps)	解密速度 (Mbps)
256 Bytes	uBlock-128/128	66.1	60.2
	uBlock-128/256	48.8	45.5
	uBlock-256/256	38.6	35.3
1 MB	uBlock-128/128	168	163.2
	uBlock-128/256	127	121.2
	uBlock-256/256	103.9	93.1

4.3 硬件性能

为说明 uBlock 算法的硬件实现性能, 我们利用 Verilog 语言实现了 uBlock, 与算法设计一致, 采用以轮为单位的迭代实现, 测试了硬件仿真实现的性能, 如表 15 所示. 测试环境为: 仿真工具 ModelSim SE, 综合工具 Sysnopsys Design Compiler, 工艺元件库为 TSMC 90 nm tcbn90ghpwc_ccs.

采用 32 个分组数据进行加/解密, 只调用 1 次密钥扩展算法. 算法实现的加/解密运算用时包含密钥扩展运算用时 (单位: 微秒). 算法电路面积规模包含加密、解密、密钥扩展运算的算法整体实现的含互连线面积 (单位: 平方微米). 速率单位 Mbps.

表 **15** uBlock 算法的硬件实现性能

Table 15 Hardware implementation performance of uBlock

硬件实现性能	**uBlock-128/128**	**uBlock-128/256**	**uBlock-256/256**
运算周期	576	832	832
时钟约束	1.38 ns	1.32 ns	1.25 ns
加密速率	5152.9 Mbps	3729.6 Mbps	7876.9 Mbps
解密速率	4996.8 Mbps	3616.6 Mbps	7638.2 Mbps
面积	45 767 um^2	54 555 um^2	86 992 um^2
等效面积	16 216 GE	19 329 GE	30 822 GE
加密吞面比	0.113	0.068	0.091
解密吞面比	0.109	0.066	0.088

此外, 考虑到 CTR 等工作模式不需要分组密码的解密运算, 我们测试了 uBlock 在单独加密模式下的硬件实现性能, 如表 16 所示. 采用以轮为单位的迭代实现, 包含加密和密钥扩展算法. 类似地, 采用 32 个分组数据进行加密, 只调用 1 次密钥扩展算法. 算法实现的加密运算用时包含密钥扩展运算用时 (单位: 微秒). 算法电路面积规模包含加密、密钥扩展运算的算法整体实现的含互连线面积 (单位: 平方微米). 速率单位 Mbps.

4.4 其他平台和应用场景

uBlock 算法适宜轻量化实现, 可适用于资源受限环境. uBlock 算法采用了 4 比特 S 盒, 和 8 比特 S 盒相比, 不仅硬件实现代价小, 而且延迟、能耗等方面都有明显优势. uBlock 算法的 S 盒仅需要 2 个与非门、2 个或非门、2 个异或门、2 个异或非门, 关键路径为 4. uBlock 算法的线性层采用了简单的循环移位、异或、字节置换运算, 具有很好的硬件实现性能. 密钥扩展采用随用即生成的方式生成轮密钥, 不增加存贮的负担; 常数利用 8 比特线性反馈移位寄存器产生, 硬件实现代价小.

表 **16** uBlock 算法的加密硬件实现性能

Table 16 Encryption hardware implementation performance of uBlock

加密硬件实现性能	uBlock-128/128	uBlock-128/256	uBlock-256/256
运算周期	576	832	832
时钟约束	1.25 ns	1.20 ns	1.20 ns
加密速率	5688.9 Mbps	4102.6 Mbps	8205.1 Mbps
面积	24 517 um^2	25 729 um^2	41 062 um^2
等效面积	8687 GE	9116 GE	14 549 GE
加密吞面比	0.232	0.159	0.200

uBlock 算法适宜低延迟实现. 基于差分和线性等密码学性质最优的 16 个仿射等价类, 精心构造了关键路径短、面积小的 4 比特 S 盒. 同时, 线性层大量使用了循环移位、字节置换等拉线操作, 极大的降低了加密算法每轮的时延. 在此基础上, 针对具体应用的时钟约束条件, 可以通过以算法的 2 (或 4) 轮作为单位, 每个时钟周期内完成 2 (或 4) 轮加密运算, 实现满足低延迟应用环境的需求.

uBlock 算法适用于 8 位微处理器实现. uBlock 算法的主要运算仅包含 4 比特 S 盒、以 4 比特/8 比特为单位的置换、以及异或等逻辑运算, 非常适合 8 位微处理器实现. 通过将 2 个 4 比特 S 盒并置看作一个 8 比特 S 盒, 整个非线性层可以按 8 比特字节查表的方式简单实现, 存储 8 比特 S 盒仅需要 256 字节 RAM. 线性层中按 8 比特为单位的置换, 可以通过寄存器直接操作, 无需额外指令; 按 4 比特为单位的置换, 可以通过 swap 指令高效实现.

uBlock 算法具有良好的防护侧信道攻击的密码特性. 门限实现掩码方案是当前主流的侧信道防护技术, 在此以它为代表说明 uBlock 算法防护的低实现代价特性. 一个门限防护方案就是基于门限计算和秘密共享的思想, 分解输入变元和相应的函数输出, 从而使得整个算法执行满足正确性、非完备性和均匀性. 针对一个特定的函数, 构造一个门限实现, 满足输入均匀性、正确性和非完备性是容易的并且是直接的, 但是其函数均匀性并不能总被满足, 为满足该性质常常需要添加新的随机数. 而 uBlock 算法在一阶门限实现中无需添加新的随机数, 这将大幅降低其防护代价. 对于密码学性质最优的 4 比特 S 盒, 文献 [10] 通过函数分解等技术给出了相应的门限实现方案, uBlock 算法的 S 盒属于最易门限实现的类别, 可以分解为二次 F 置换和 G 置换, 并且对于某些 F 置换和 G 置换可以直接找到满足函数均匀性的 3-share 的门限实现, 其一阶门限实现的实现面积很低, 并且在算法执行过程中无需增加新的随机数.

5 结束语

uBlock 算法结构简单清晰, 不同参数的算法基于同一框架, 算法便于扩展; 采用 S 盒和分支数的理念, 针对典型分析方法具有可证明安全性. 简单的设计使得 uBlock 算法的安全性分析评估相对容易, 可利用自动分析技术评估其对差分分析、线性分析、积分分析、不可能差分分析、中间相遇攻击等分析方法的安全性. 目前的评估结果显示 uBlock 算法针对现有分析方法具有足够的安全冗余.

uBlock 算法适应各种软硬件平台. PC 上可以利用 SSE 和 AVX2 等指令集高效实现; 对于主流的 ARM 平台, 可以利用 neon 指令集优化实现; 非常适合 8 位微处理器实现. 硬件实现方面, 并行结构、模块化、简单的 S 盒和线性变换, 使得 uBlock 算法的硬件实现简单而有效, 既可以高速实现, 保障高性能环境的安全应用, 也可以轻量化实现, 满足资源受限环境的安全需求.

uBlock 算法的设计目标是安全性高、可扩展性好、适应性强, 以满足多个行业领域对分组密码算法的应用需求. 欢迎国内外专家学者分析评估 uBlock 算法的安全性, 研究 uBlock 算法在各种应用环境的优化实现技术.

致谢

uBlock 算法硬件实现性能评估得到清华大学苏冠通博士的大力支持, 在此表示感谢.

References

[1] GUO Z, WU W, GAO S. Constructing lightweight optimal diffusion primitives with Feistel structure[C]. In: Selected Areas in Cryptography—SAC 2015. Springer Berlin Heidelberg, 2016: 352–372. [DOI: 10.1007/978-3-319-31301-6_21]

[2] BIHAM E, SHAMIR A. Differential Cryptanalysis of the Data Encryption Standard[M]. Springer New York, 1993: 1–188. [DOI: 10.1007/978-1-4613-9314-6]

[3] AOKI K, ICHIKAWA T, KANDA M, et al. Camellia: A 128-bit block cipher suitable for multiple platforms—Design and analysis[C]. In: Selected Areas in Cryptography—SAC 2000. Springer Berlin Heidelberg, 2000: 39–56. [DOI: 10.1007/3-540-44983-3_4]

[4] SHIRAI T, SHIBUTANI K, AKISHITA T, et al. The 128-bit block cipher CLEFIA (extended abstract)[C]. In: Fast Software Encryption—FSE 2007. Springer Berlin Heidelberg, 2007: 181–195. [DOI: 10.1007/978-3-540-74619-5_12]

[5] WU W L, ZHANG L. LBlock: A lightweight block cipher[C]. In: Applied Cryptography and Network Security—ACNS 2011. Springer Berlin Heidelberg, 2011: 327–344. [DOI: 10.1007/978-3-642-21554-4_19]

[6] MATSUI M. Linear cryptanalysis method for DES cipher[C]. In: Advances in Cryptology—EUROCRYPT '93. Springer Berlin Heidelberg, 1994: 386–397. [DOI: 10.1007/3-540-48285-7_33]

[7] ZHANG H, WU W L. Structural evaluation for generalized Feistel structures and applications to LBlock and TWINE[C]. In: Progress in Cryptology—INDOCRYPT 2015. Springer Berlin Heidelberg, 2015: 218–237. [DOI: 10.1007/978-3-319-26617-6_12]

[8] TODO Y, MORII M. Bit-based division property and application to SIMON family[C]. In: Fast Software Encryption—FSE 2016. Springer Berlin Heidelberg, 2016: 357–377. [DOI: 10.1007/978-3-662-52993-5_18]

[9] XIANG Z, ZHANG W, BAO Z, et al. Applying MILP method to searching integral distinguishers based on division property for 6 lightweight block ciphers[C]. In: Advances in Cryptology—ASIACRYPT 2016, Part I. Springer Berlin Heidelberg, 2016: 648–678. [DOI: 10.1007/978-3-662-53887-6_24]

[10] BILGIN B, NIKOVA S, NIKOV V, et al. Threshold implementations of all 3×3 and 4×4 S-boxes[C]. In: Cryptographic Hardware and Embedded Systems—CHES 2012. Springer Berlin Heidelberg, 2012: 76–91. [DOI: 10.1007/978-3-642-33027-8_5]

作者信息

吴文玲 (1966–), 博士, 研究员. 主要研究领域为密码学.
wwl@tca.iscas.ac.cn

张蕾 (1981–), 博士, 副研究员. 主要研究领域为对称密码.
zhenglei@tca.iscas.ac.cn

郑雅菲 (1988–), 博士. 主要研究领域为分组密码.
zhengyafei@tca.iscas.ac.cn

李灵琛 (1988–), 博士. 主要研究领域为分组密码.
lilingchen@tca.iscas.ac.cn