Raindrop 系列分组密码算法

设计文档

王美琴 李永清 李木舟 付勇 樊燕红 黄鲁宁

山东大学网络空间安全学院

1 算法描述

1.1 算法参数

本文档给出了一种新型分组密码算法—Raindrop。该算法共包含 3 种版本，Raindrop-128-128、Raindrop-128-256 以及 Raindrop-256-256，每个版本的参数详见表 1。

表 1：Raindrop 算法的三种版本

版本	分组长度	密钥长度	轮数
Raindrop-128-128	128	128	60
Raindrop-128-256	128	256	80
Raindrop-256-256	256	256	100

1.2 算法结构

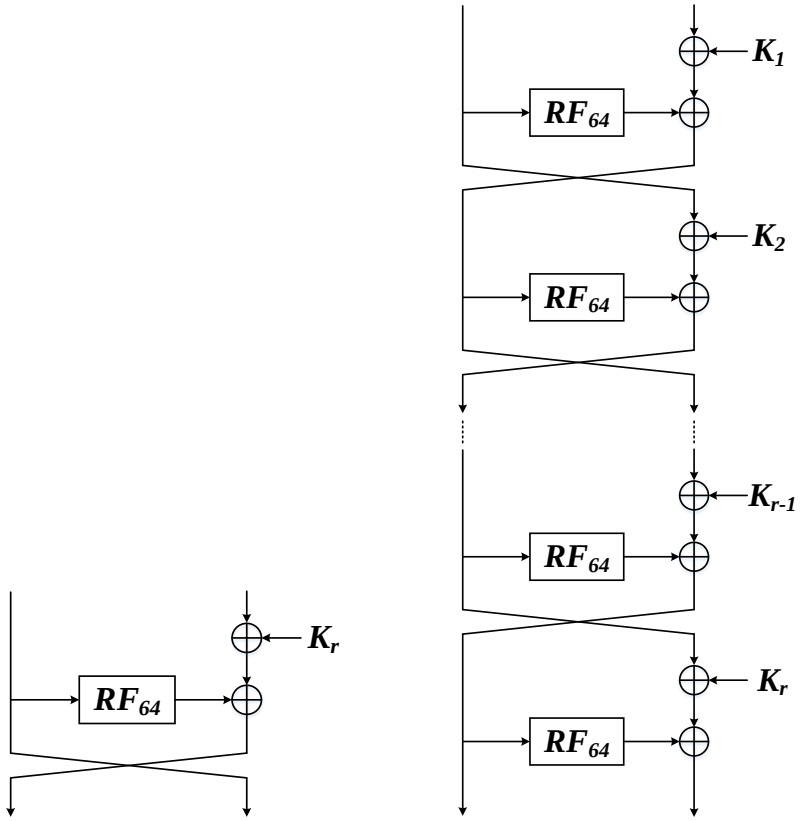
1.2.1 分组长度为 128 比特的算法结构

Raindrop-128-128/256 算法采用两支的 Feistel 结构，对 128 比特长度的明文进行操作。将 128 比特的明文从右到左标号，序号从 0 开始至 127。128 比特的明文在运算时被分成左右两支，第 64 到 127 比特作为左支的 64 比特的状态值，第 0 到 63 比特作为右支的 64 比特的状态值。算法 1 轮的结构如图 1(a)所示，右支 64 比特异或 64 比特轮密钥K_r，与通过轮函数RF_{64}的左支进行异或，然后交换左右支，注意算法最后一轮不进行交换。其中RF_{64}是 Raindrop-128-128/256 的轮函数，包含 S 盒（$Sbox$）、行混合（$MixRow$）和比特级循环移位（$BitRot$）三种操作，加密算法的整体结构如图 1(b)所示，解密时只需将密文作为“明文”输入到加密流程中，但是使用的轮密钥的顺序反序，即依次使用$K_r, K_{r-1}, \cdots, K_1$作为每一轮的轮密钥。

算法的轮函数RF_{64}对 64 比特的状态进行操作，此 64 比特的状态可以表示成4×4的二维数组，称作状态矩阵。给定 64 比特的状态$p_0 p_1 \ldots p_{15}$，可以将其按照如下顺序映射成4×4的状态矩阵：

$$\begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{pmatrix},$$

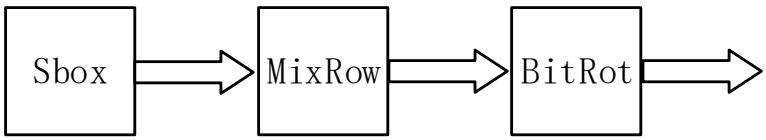
其中p_i的长度$|p_i|$满足：当$i = 0,4,8,12,2,6,10,14$时，有$|p_i| = 3$；其余情况下，有$|p_i| = 5$。每一个p_i在本文中被称为一个 word，因此 64 比特的状态可以分成 16 个 word 的组合。



(a) 1 轮结构　　(b) 整体结构

图 1：Raindrop-128-128/256 加密算法

轮函数RF_{64}对此状态矩阵依次利用三种操作进行更新： S 盒（$Sbox$）、行混合（$MixRow$）和比特级循环移位（$BitRot$）。



1. S 盒（$Sbox$）

算法采用两种不同大小的 S 盒，分别是 3 比特和 5 比特 S 盒。两种 S 盒的构造方式是相同的，均是采用 Keccak 的 S 盒的构造方式，如图 2 所示。其中符号

“$\forall$”代表比特级取反运算，“$\cup$”代表比特级与运算，“$\oplus$”代表比特级异或运算。对于k比特的 S 盒，假设输入为$(x_{k-1}, x_{k-2}, \cdots, x_1, x_0)$，输出为$(y_{k-1}, y_{k-2}, \cdots, y_1, y_0)$，则

$$y_i = x_i \oplus \left(\sim x_{(i-1)\ mod\ k}\right) \& x_{(i-2)\ mod\ k}$$

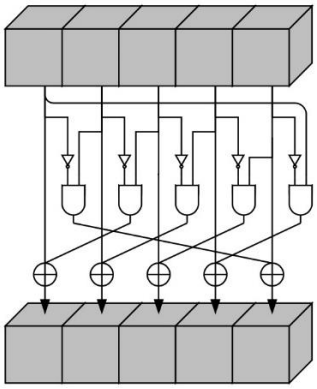


图 2：Keccak 的 5 比特 S 盒的构造方式

2. 行混合（$MixRow$）

行混合矩阵为

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix},$$

此矩阵右乘状态矩阵，可以得到行混合之后的值，即：

$$\begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{pmatrix} \times \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}$$

$$\Rightarrow \begin{pmatrix} p_4 \oplus p_{12} & p_8 \oplus p_{12} & p_0 & p_0 \oplus p_4 \\ p_5 \oplus p_{13} & p_9 \oplus p_{13} & p_1 & p_1 \oplus p_5 \\ p_6 \oplus p_{14} & p_{10} \oplus p_{14} & p_2 & p_2 \oplus p_6 \\ p_7 \oplus p_{15} & p_{11} \oplus p_{15} & p_3 & p_3 \oplus p_7 \end{pmatrix}$$

3. 比特级循环移位（$BitRot$）

记状态矩阵的第i列的级联值为$Col_i = p_{4i} \parallel p_{4i+1} \parallel p_{4i+2} \parallel p_{4i+3}$，其中$0 \leq i \leq 3$，对每一列按照下述方式进行比特级的循环移位，即：$Col_0$不变，$Col_1$向左循环 6 比特，$Col_2$向左循环 7 比特，$Col_3$向左循环 12 比特。

1.2.2 分组长度为 256 比特的算法结构

Raindrop-256-256 算法也采用两支的 Feistel 结构，其使用与 Raindrop-128-128/256 算法中使用的RF_{64}类似结构的轮函数RF_{128}，但RF_{128}处理 128 比特的数据。算法对 256 比特长度的明文进行操作，将明文分成左右两支，每支 128 比特。256 比特明文的第 128 到 255 比特作为左支的 128 比特的状态值，第 0 到 127 比特作为右支的 128 比特的状态值。右支 128 比特异或 128 比特轮密钥K_r，与通过轮函数RF_{128}的左支进行异或，然后交换左右支，注意算法最后一轮不进行交换。加密算法的整体结构如图 3 所示，解密时只需将密文作为“明文”输入到加密流程中，但是使用的轮密钥的顺序反序，即依次使用$K_r, K_{r-1}, \cdots, K_1$作为每一轮的轮密钥。

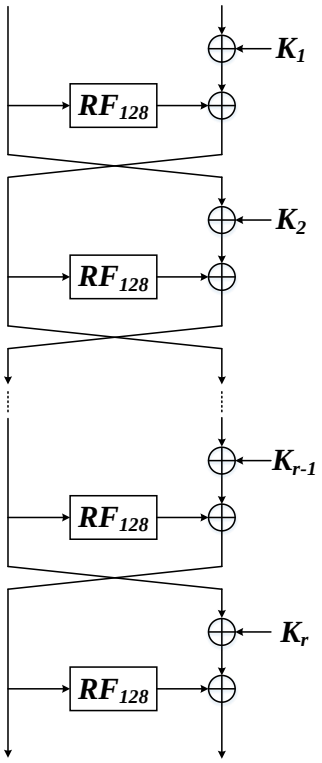


图 3：Raindrop-256-256 加密算法的整体结构

算法的轮函数RF_{128}对 128 比特的状态进行操作，此 128 比特的状态可以表示成4×4的二维数组，称作状态矩阵。给定 128 比特的状态$p_0 p_1 \ldots p_{15}$，可以将其按照如下顺序映射成4×4的状态矩阵：

$$\begin{pmatrix} p_0 & p_4 & p_8 & p_{12} \\ p_1 & p_5 & p_9 & p_{13} \\ p_2 & p_6 & p_{10} & p_{14} \\ p_3 & p_7 & p_{11} & p_{15} \end{pmatrix},$$

其中p_i的长度$|p_i|$满足：当$i = 0,4,8,12,2,6,10,14$时，有$|p_i| = 7$；其余情况下，有$|p_i| = 9$。与RF_{64}类似，RF_{128}对此状态矩阵依次利用三种操作进行更新： S 盒（$Sbox$）、行混合（$MixRow$）和比特级循环移位（$BitRot$）。

1. S 盒（$Sbox$）

与 Raindrop-128-128/256 算法类似，Raindrop-256-256 算法采用两种不同大小的 S 盒，分别是 7 比特和 9 比特 S 盒。S 盒的构造方式采用 Keccak 的 S 盒的构造方式。

2. 行混合（$MixRow$）

行混合矩阵为

$$\begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix},$$

与 128 分组类似，此矩阵右乘状态矩阵，可以得到行混合之后的值。

3. 比特级循环移位（$BitRot$）

记状态矩阵的第i列的级联值为$Col_i = p_{4i} \| p_{4i+1} \| p_{4i+2} \| p_{4i+3}$，其中$0 \le i \le 3$，对每一列按照下述方式进行比特级的循环移位：$Col_0$不变，$Col_1$向左循环 12 比特，$Col_2$向左循环 14 比特，$Col_3$向左循环 24 比特。

1.3 密钥生成算法

1.3.1 Raindrop-128-128 的密钥生成算法

假设我们需要生成R轮的轮密钥，那么首先将 128 比特的主密钥赋给密钥状态TK_1，然后按照图 4 所示的方式依次生成密钥状态TK_r，其中r代表轮数，满足$2 \le r \le R$，$r||0 \dots 0$是密钥生成算法中使用的轮常数，$0 \dots 0$代表 64 比特的“0”，即轮常数异或在TK_r^3上，A^4代表执行连续的 4 次 A 函数。在执行 A 函数时，将 128 比特的密钥状态分成 8 块，每块 16 比特，即$TK_r = TK_r^0 \| TK_r^1 \| TK_r^2 \| TK_r^3 \| TK_r^4 \| TK_r^5 \| TK_r^6 \| TK_r^7$。A 函数具体的形式见图 5，其中符号“≫”代表右移操作，

“⋘”代表左循环移位操作。注意函数的最后一步是对这 128 比特的状态进行一个整体的左循环移 5 位的操作。第r轮的 64 比特的轮密钥K_r是TK_r的第 64 至 127 比特，即$K_r = TK_r[127:64]$。

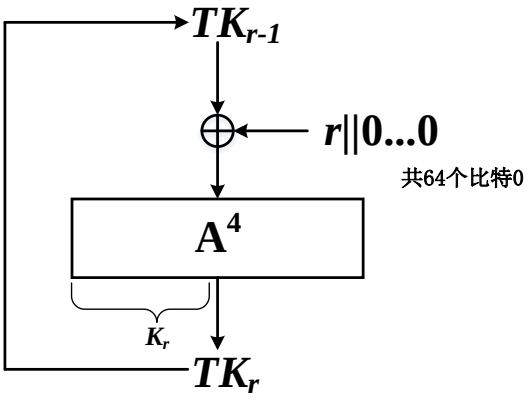


图 4：Raindrop-128-128 的密钥生成算法

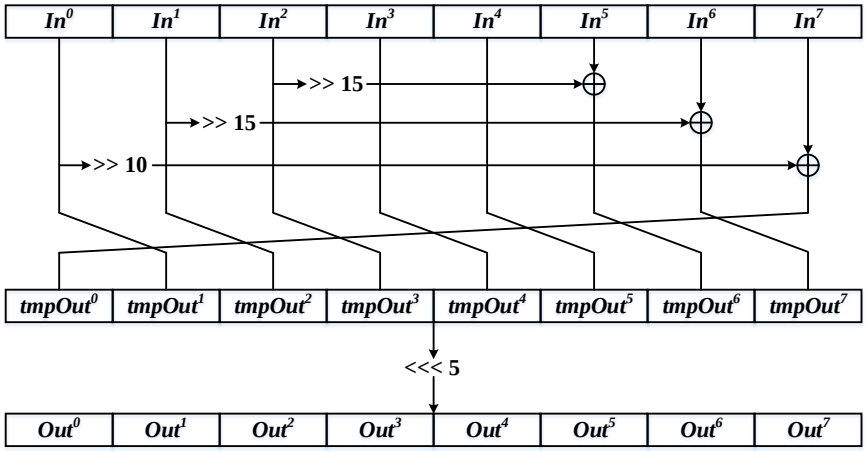


图 5：A 函数

1.3.2 Raindrop-128-256 的密钥生成算法

Raindrop-128-256 算法的密钥生成算法的整体结构与 Raindrop-128-128 类似，首先将 256 比特的主密钥赋给密钥状态TK_1，然后按照图 6 所示的方式依次生成密钥状态TK_r，其中r代表轮数，满足$2 \le r \le R$，$r||0 \dots 0$是密钥生成算法中使用的轮常数，$0 \dots 0$代表 160 比特的“0”，即轮常数异或在TK_r^2。B^4代表执行连续的 4 次 B 函数。在执行 B 函数时，将 256 比特的密钥状态分成 8 块，每块 32 比特，即$TK_r = TK_r^0 \| TK_r^1 \| TK_r^2 \| TK_r^3 \| TK_r^4 \| TK_r^5 \| TK_r^6 \| TK_r^7$。B 函数具体的形式见图 7，其中符号“≫”代表右移操作，“≪”代表左移操作，“⋘”代表左循环移位

操作。注意函数的最后一步是对这 256 比特的状态进行一个整体的左循环移 13 位的操作。第r轮的 64 比特的轮密钥K_r是TK_r的第 240 至 255 比特，第 208 至 223 比特，第 166 至 181 比特，第 144 至 159 比特的级联值，即：$K_r = TK_r[255:240] \| TK_r[223:208] \| TK_r[181:166] \| TK_r[159:144]$。

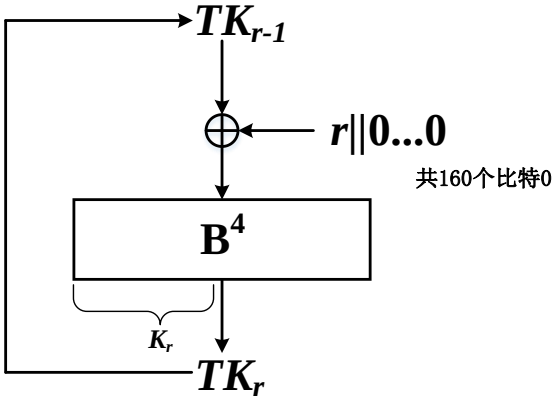


图 6：Raindrop-128-256 的密钥生成算法

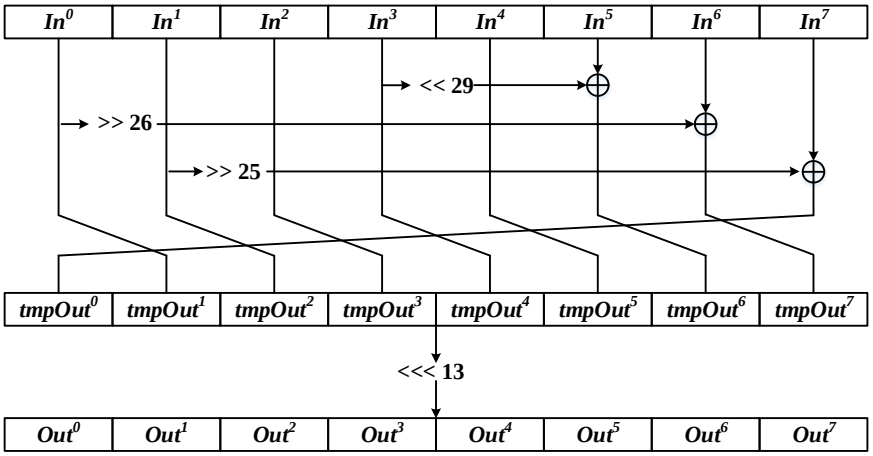


图 7：B 函数

1.3.3 Raindrop-256-256 的密钥生成算法

Raindrop-256-256 的密钥生成算法与 Raindrop-128-256 算法的密钥生成算法相同，只不过 Raindrop-256-256 使用 128 比特的轮密钥。第r轮的 128 比特的轮密钥K_r是TK_r的第 128 至 255 比特，即：$K_r = TK_r[255:128]$。

2 设计原理

在设计过程中，主要考察硬件实现的效率。本分组密码采取类 Keccak 非线

性函数作为 S 盒，主要原因是延时短，软件实现抵抗测信道攻击。同时二元域是深度为 1 的矩阵，降低了延时，为了便于分析，我们采取线性的密钥生成算法，通过搜索扩散性来选取参数，在异或数较少的情况下，使得扩散速度较快。

3 安全性分析

本节分别从差分分析和线性分析的角度对 Raindrop 算法三种版本的安全性进行深入的研究。在研究过程中，我们使用基于 STP 搜索工具构建的自动化搜索模型来寻找相应的区分器，对我们研究算法的安全性起到了有效的帮助。

3.1 差分分析

在一条差分路线中，如果 S 盒的输入差分不是 0，则其被称为活性 S 盒。显然，如果差分路线中包含的活性 S 盒的个数越少，其路线概率越高。因此最小的活性 S 盒个数可以用来衡量分组密码算法抵抗差分分析的能力。

利用 STP 搜索工具建立的自动化搜索模型，我们发现 Raindrop-128-128/256 算法 16 轮最小活跃 S 盒个数为 34，Raindrop-256/256 算法 16 轮最小活跃 S 盒个数大于等于 32。

根据 S 盒的差分分布表，我们可知 3 比特、5 比特、7 比特、9 比特 S 盒的非 0 差分传播概率的最大值均为2^{-2}，所以对于分组长度为n比特的算法而言，当最小活性 S 盒的个数大于$n/2$时（其中n为分组长度），将不存在有效的差分路线。因此 Raindrop-128-128/256 算法不存在 32 轮及更长轮数的有效的差分路线。由于 Raidrop-256-256 算法 16 轮的最小活性 S 盒的个数大于等于 32，因此其 32 轮的最小活性 S 盒的个数将大于等于 64，64 轮的最小活性 S 盒的个数将大于等于 128，即不存在 64 轮及更长轮数的有效的差分路线。

因此，Raindrop 算法是抵抗差分攻击的，同时，我们认为 Raindrop 算法设置的轮数也是足够抵抗相关密钥分析的。

3.2 线性分析

与差分分析类似，在一条线性路线中，如果 S 盒的输出掩码不是 0，则其被称为活性 S 盒。同时，如果线性路线中包含的活性 S 盒的个数越少，路线的相关

性越大。因此最小的活性 S 盒个数可以用来衡量分组密码算法抵抗线性分析的能力。

我们搜索发现 Raindrop-128-128/256 算法 16 轮最小活跃 S 盒个数等于 32，Raindrop-256-256 算法 16 轮最小活跃 S 盒个数等于 32。

记此条线性路线的偏差是Cor^{-2}，则此条线性路线去做密钥恢复时所需要的数据复杂度大概是Cor^{-2}。因此当$Cor^{-2} < 2^n$时，这条相关路线才被称作有效路线。根据 S 盒的线性近似表，我们可知 3 比特、5 比特、7 比特、9 比特 S 盒的非 0 掩码传播相关性的最大值均为2^{-1}。因此当路线中包含的最小活性 S 盒的个数大于$n/2$时（其中n为分组长度），将不存在有效的线性路线。因此，对于 Raindrop-128-128/256 而言，其不存在 32 轮及更长轮数的有效的线性路线。由于 Raidrop-256-256 算法 16 轮的最小活性 S 盒的个数等于 32，因此不存在 64 轮及更长轮数的有效的线性路线。

因此，Raindrop 算法是抵抗线性攻击的。

3.3 不可能差分分析

不可能差分分析是差分分析的一个变种，不可能差分分析与经典差分分析不同的地方在于，经典的差分分析利用高概率差分来回复密钥，而不可能差分分析是寻找一条不可能的差分路径，即这条差分路径的概率为 0，从而排除那些导致概率为 0 的差分出现的候选密钥。

对于 Raindrop-128-128/256 和 Raindrop-256-256 算法来说，均不存在 13 轮及以上的 1-1-word 的不可能差分。

因此，Raindrop 算法是抵抗 1-1-word 不可能差分的。

3.5 积分分析

积分攻击可以看作差分攻击的一种推广，积分攻击将状态进行加和，通过分析特定比特位上的元素出现次数的奇偶性来确定该比特位上所有值的异或值，从而判断该位置比特的平衡性。积分攻击不需要对密钥进行计数，淘汰不通过检测的密钥来达到攻击的目的。

Raindrop-128-128/256 积分区分器最长为 7 轮，Raindrop-256-256 积分区分器

最长为 13 轮。

因此 Raindrop 算法是抵抗积分攻击的。

3.6 零相关分析

作为线性分析的一种，零相关线性分析与不可能差分分析的思想一致，其利用相关度为 0 的线性路线来做密钥恢复攻击。

在限定输入掩码和输出掩码各只有一个 word 活跃的前提下，对于 Raindrop-128-128/256 算法，我们找到的最长的零相关线性路线是 10 轮，对于 Raindrop-256-256 算法，我们找到的最长的零相关线性路线是 12 轮。

因此，Raindrop 算法是抵抗零相关分析的。

4 性能分析

4.1 软件性能分析

本节为 Raindrop 算法在 64bit Windows 环境和 32bit ARM 环境下的实现结果。

测试环境分别为：Intel(R) Core(TM) i7-6700 CPU 2.40 GHz 主频、64 位 Windows10 操作系统、8GB 内存，X86-64 环境和基于 STM32 F103 的开发板(主频 72Mhz)：stm32f1zet6 核心板 6、512K 片内 Flash、64K 片内 RAM。

开发工具：VS2017、X64 avx2 Release

测试方法：采用 256 字节数据作为输入进行性能测试，每次测试变换密钥，取 10 5 次 CBC 模式运算测试结果的平均值为加/解密速率（单位：Mbps）的结果。

加密速率=加密数据大小/加密执行时间（单位：Mbps）

解密速率=解密数据大小/解密执行时间（单位：Mbps）

其中加/解密执行时间应包含密钥扩展算法运算时间。

下表为 Raindrop 算法软件实现自测试结果记录。

表 3：Raindrop 算法软件实现自测试结果记录

算法软件实现类别		加密速率	解密速率
64bit Windows 环境下的算法速率优化 C 实现	128/128	242Mbps	3126Mbps
	128/256	181Mbps	2534Mbps
	256/256	271Mbps	1892Mbps
32bit ARM 环境下的算法速率优化 C 实现	128/128	1.19Mbps	1.19Mbps
	128/256	0.91Mbps	0.91Mbps
	256/256	0.98Mbps	0.98Mbps

注：解密速率是基于多路实现的结果。

4.2 硬件性能分析

本节给出 Raindrop 算法的 Verilog 硬件仿真实现结果。

采用的仿真工具为 ModelSim SE 10.1a，综合工具为 Synopsys Design Vision（F-2011.09-SP1Version），以及工艺库为 HJTC 0.11um。算法 Verilog 硬件仿真实现采用 32 个分组数据进行加/解密。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）。算法电路面积规模为包含加密、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）。

具体方法如下：

（1） 仿真验证

使用 ModelSim 对算法实现正确性进行仿真验证，并记录 32 个分组数据运算（含密钥扩展运算）占用的时钟周期数。

（2） 性能自测试

在给定的时钟约束条件上进行前端综合，得到并记录以下测试数据：

1) 关键路径时长

在 Synopsys Design Complier 中进行关键路径时长测试。根据算法设计提供的自测数据，设置相应的时钟约束条件，通过综合软件得到关键路径时长。

2) 速率

在关键路径时长合理的条件下，通过以下公式计算加解密速率：

加密速率=加密数据长度/（时钟约束条件*加密占用的时钟钟周期数）（单位：Mbps）

解密速率=解密数据长度/（时钟约束条件*解密占用的时钟周期数）（单位：Mbps）

3) 面积

在可满足的最小时钟约束条件下，在 HJTC 0.11um 工艺库基础上进行综合，记录算法实现面积（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）。

4) 吞面比

加密吞面比=加密速率（单位：Mbps）/算法电路面积（单位：平方微米）

解密吞面比=解密速率（单位：Mbps）/算法电路面积（单位：平方微米）

以下是 Raindrop 算法 Verilog 硬件仿真实现自测试结果记录：

表 3：Raindrop 算法 Verilog 硬件仿真实现自测试结果记录

	自测试结果		
	128/128	128/256	256/256
运算周期	3091	5201	6501
时钟约束	2.18 ns	2.1 ns	2.2 ns
加密速率	978.46Mbps	738.54Mbps	1163.64Mbps
解密速率	948.46Mbps	738.54Mbps	1128.03Mbps
面积	26672.28um^2	42137.82um^2	50238.28um^2
面积	2750.00GE	4344.55GE	5179.74GE
加密吞面比	0.036690 Mbps/ um^2	0.018081 Mbps/um^2	0.023162 Mbps/um^2
解密吞面比	0.035560 Mbps/ um^2	0.017526 Mbps/um^2	0.022453 Mbps/um^2

5 优缺点声明

优势说明

通过分析 Raindrop 算法的安全性、软件性能评测、硬件实现等指标，具体优点及缺点如下。

5.1 优点一：安全性强

Raindrop 算法采用 3、5、7、9 比特 S 盒，四种 S 盒的最大差分概率和最大线性偏差均为2^{-2}，四种 S 盒最小差分概率和最小线性偏差分别为2^{-3}、2^{-5}、2^{-7}、2^{-9}。并且在进行差分分析和线性分析的安全性测试时，均采用活跃 S 盒个数进行评测，因此，在给定轮数下的 Raindrop 算法可以认为是足够安全的。

5.2 优点二：易于门限实现

轮函数使用的操作均是 AND 和 XOR 操作，容易进行门限实现，且实现代价很小。

5.3 优点四：加解密一致

采用 Feistel 结构，通过两部分加密状态的交与异或，换使得加密过程与解密过程一致，能够节省较多的硬件资源，不需要为解密算法付出额外代价。

5.4 优点三：易于轻量化实现

在 Raindrop 算法的轮函数中，非线性运算只包括 1 个与运算、1 个异或运算和 1 个非运算，且线性层的二元域矩阵深度为 1，因此硬件面积较小，易于轻量化实现。

5.5 缺点：软件实现稍有繁琐，安全界评测方法可以改进

由于 Raindrop 算法采用了 3、5、7、9 比特 S 盒，使得软件实现较为繁琐，同时 Raindrop 算法的独特性，在安全性评测时主要对活跃 S 盒的个数进行评估。最终使得 Raindrop 算法的安全界较松，因此降低 Raindrop 算法的轮数仍有很大潜力。