doi:10.3969/j.issn.1000-436x.2012.11.012

构造零和区分器的新方法

董乐[1,2]，吴文玲[1]，吴双[1]，邹剑[1,2]
(1. 中国科学院 软件研究所，北京 100190；2. 中国科学院 研究生院，北京 100190)

摘 要：通过分析具有相似结构的 AES 类置换的扩散性质，提出了一种构造零和区分器的新方法。这种方法组合了高阶积分攻击和高阶差分攻击，利用选择的一个确定其活跃模式的中间状态，构造一条高阶积分路径，然后以此路径的 2 个终点作为起始点，再构造高阶差分路径。利用此方法，改进了对 PHOTON 杂凑函数族 2 个置换的全轮零和攻击，并对进入 SHA-3 最终轮的 JH 算法的核心函数构造了 31.5 轮的零和区分器。

关键词：AES 类；零和区分器；高阶差分攻击；高阶积分攻击；PHOTON；JH
中图分类号：TN918 **文献标识码**：A **文章编号**：1000-436X(2012)11-0091-09

Novel method of constructing the zero-sum distinguishers

DONG Le[1,2], WU Wen-ling[1], WU Shuang[1], ZHOU Jian[1,2]
(1. Institute of Software, Chinese Academy of Sciences, Beijing 100190, China;
2. Graduate University, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: A novel method of constructing the zero-sum distinguishers for AES-like permutations was proposed by considering the diffusion properties of these permutations, which have the similar construction. The method combined the higher-order integral attack and the higher-order differential attack. Utilizing the selected intermediate-state-structure whose active mode was determined, a higher-order integral path was constructed. Then, a higher-order differential trace was built from the two ends of the integral path. Applying the method, the full-round zero-sum attack on two permutations adopted by the PHOTON family was improved. Besides, a 31.5-round zero-sum distinguisher of the core function of JH hash function was constructed, which entered into the final round of the SHA-3 competition.

Key words: AES-like; zero-sum distinguishers; higher-order differential attack; higher-order integral attack; PHOTON; JH

1 引言

2001 年，AES[1]被美国国家标准技术研究所（NIST）颁布为新的加密标准，其结构被密码工作者广为研究。随着 RFID 标签、智能卡和无线传感器等智能设备的推广，轻量级杂凑函数的设计成为近年热点之一。在 2011 年美洲密码年会上发布的、采用类似 AES 结构置换的 PHOTON[2, 3]族杂凑函数就是其中的一个。而在近年来举办的 SHA-3 竞赛中，很多算法也都有和 AES 相似的结构或者设计理念，包括进入最终轮的 JH[4]算法，其核心置换就采用了多维 AES 设计理念。这些与 AES 有相似结构的置换通常被称为 AES 类置换。

PHOTON 杂凑函数族是由 J Guo 等设计，它采用了 AES 类置换。设计者在设计文档中给出了几种区分器攻击[2]。其中反弹区分器可以攻击到 8 轮，积分区分器可以攻击到 7 轮。除此之外，基于对代数次数的估计，零和区分器可以对其中 4 个版本攻击到全轮 12 轮，对采用 8bit S 盒的版本攻击到 8 轮。

JH 杂凑函数是进入 SHA-3 竞赛最终轮的 5 个算

收稿日期：2011-12-09；修回日期：2012-03-30
基金项目：国家自然科学基金资助项目（61272476, 61232009）
Foundation Item: The National Natural Science Foundation of China (61272476, 61232009)

法之一，由学者 H Wu 独立设计。该杂凑函数采用了一种新的压缩函数结构。除此之外，在其轮函数的设计中，还采用了广义 AES 的设计理念。对于 JH 算法，V Rijmen 等利用反弹攻击的思想，构造了杂凑函数 16 轮的半自由起始碰撞，和 22 轮压缩函数的半自由起始近似碰撞[5]。在 2010 年的 SHA-3 第 2 次会议上，M S Turan 等以可实现的 $2^{23.24}$ 的时间复杂度，构造了 10 轮压缩函数的半自由起始近似碰撞[6]。在 2011 年的美洲密码年会上，N-Plasencia 将 V Rijmen 等的 22 轮半自由起始近似碰撞的时间和空间复杂度均降低至 2^{95}[7]。除此之外，在文献[8]中，C Boura 等对 JH 置换的代数次数进行了新的估计，得出 8 轮置换的代数次数至多为 409，PHOTON2 个置换和 JH 的现有攻击比较如表 1 所示。

1994 年，来学嘉教授提出了离散函数“高阶微分”的概念[9]，此后高阶差分[10]便成为分组密码分析的一个有力的工具。2009 年，学者 J P Aumasson 基于对 3 个 SHA-3 候选算法 KECCAK、Luffa 和 Hamsi 的区分攻击，提出了构造“零和区分器”[11]的攻击方式，这种区分器的构造便是基于算法的高阶差分性质。2010 年，C Boura 和 A Canteaut 等发现迭代置换的一些特殊性质[12]，并将其应用于 KECCAK-*f* 和 Hamsi-256。而后，C Boura 等改进了此方法，并构造出来全轮的 KECCAK-*f* 置换的零和区分器[13]。除此之外，J P Aumasson 等人还给出了 Hamsi-256 的一些“2^i-和”区分器[14]。

本文应用高阶积分攻击和高阶差分攻击，针对 AES 类置换的扩散性质，提出一种构造零和区分器的新方法。并对 PHOTON 和 JH 中的置换，构造出由高阶积分路径和高阶差分路径组成的组合路径。基于这些路径，建立了 JH 的 31.5 轮核心函数的“零和区分器”，并改进了 PHOTON 杂凑函数族中 2 个置换的“零和区分器”。

2 PHOTON 杂凑函数族与 JH 杂凑函数

本节对 PHOTON 杂凑函数族和 JH 杂凑函数进行介绍，其中主要介绍它们所使用的置换。

2.1 PHOTON 杂凑函数族

PHOTON[2, 3]杂凑函数族是由 Guo J 等人设计的一族轻量级杂凑函数，正式发布于 2011 年美洲密码年会。PHOTON 采用了 2 个主要技术：1）扩展的海绵结构；2）AES 类置换。

海绵函数是一种由固定置换构建杂凑函数的新选择。为了增加灵活性，PHOTON 采用了扩展的海绵结构。它允许每次置换 P 之后的“比特榨取”时析出比特的个数可以与“比特吸收”时吸收的比特个数不相同，这样可以减少“比特榨取”过程的时间。

PHOTON 的内部置换使用的是一个 AES 类函数。中间状态可被表示为一个 sb^2bit 大小的 $b\times b$ 的方阵，本文中称状态矩阵中的元素为“单元”，这里 s 表示每个单元的大小，即每个 S 盒的大小，b 表示每行（或者每列）包含的单元个数。PHOTON 的

表 1 PHOTON 2 个置换和 JH 的现有攻击比较

攻击目标	轮数	时间复杂度	存储	攻击类型	出处
PHOTON-160 中所用置换 P_{196}	7	2^{52}	—	积分区分器	文献[2]
	8	2^{8}	2^{4}	反弹区分器	文献[2]
	12	2^{183}	—	零和区分器	文献[2]
	12	2^{167}	—	零和区分器	本文 5.1 节
PHOTON-224 中所用置换 P_{256}	7	2^{60}	—	积分区分器	文献[2]
	8	2^{8}	2^{4}	反弹区分器	文献[2]
	12	2^{236}	—	零和区分器	文献[2]
	12	2^{223}	—	零和区分器	本文 5.2 节
JH-256 压缩函数	22	2^{156}	2^{143}	半自由起始近似碰撞	文献[5]
	22	2^{95}	2^{95}	半自由起始近似碰撞	文献[7]
	16	2^{96}	2^{96}	半自由起始碰撞	文献[7]
JH-256 杂凑函数	16	2^{178}	2^{101}	半自由起始碰撞	文献[5]
JH 压缩函数	10	2^{23}	—	半自由起始近似碰撞	文献[6]
JH 核心函数	31.5	2^{767}	—	零和区分器	本文 6.2 节

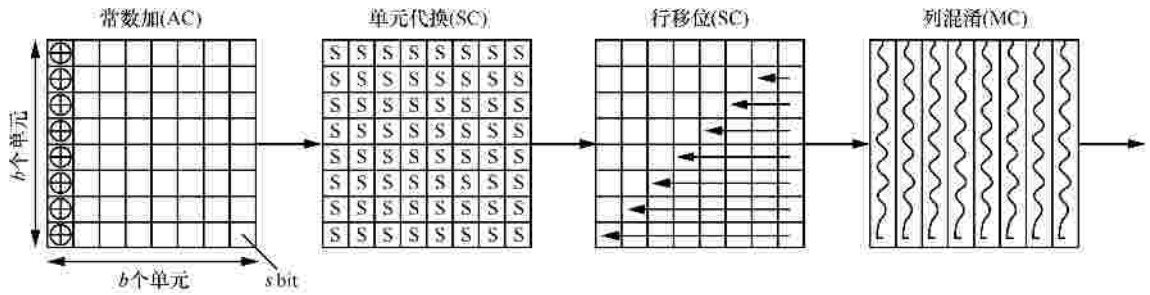


图 1　PHOTON 置换 P_{256} 的一轮

每一轮包含 4 个基本运算，如图 1 所示。

1) 常数加(AC)：状态中第一列的每个单元异或上一个常数，以破坏列之间的对称性，并区分置换的每一轮。

2) 单元代换(SC)：2 种 S 盒被用来构建置换层，PRESENT 算法中的 4bit S 盒，与 AES 算法中的 8bit S 盒。其中，8bit S 盒只用在 PHOTON 函数族的一个置换中。

3) 行移位(SR)：与 AES 的行移位非常相似。简单地说，就是第 i 行向左循环移动 i 个单元，i 从 0 开始。

4) 列混淆(MC)：为了降低实现代价，PHOTON 的混淆层使用一个含很多 0 的矩阵，并用一系列的迭代更新列向量。这个过程相当于乘以一个 MDS 矩阵。

这样，PHOTON 的轮函数可以表示为

$$R = MC \circ SR \circ SC \circ AC$$

PHOTON 杂凑函数族共包含 5 个成员，它们使用的内部置换分别为 P_{100}、P_{144}、P_{196}、P_{256} 和 P_{288}。表 2 给出这些置换内部状态大小 t、每行或者每列的单元个数 b、S 盒大小 s 和轮数 N_r。

从表中可知，除了最后一个置换，其余置换都采用了 4×4 的 S 盒。此外，每一个置换的轮数都是 12。文献[2]还给出随着轮数增加，5 个版本的置换代数次数的上界。其中 5 轮 P_{196} 和 P_{256} 置换的代数次数上界分别为 157 和 197。

表 2　　5 个内部置换的参数

版本	t	b	s	N_r
P_{100}	100	5	4	12
P_{144}	144	6	4	12
P_{196}	196	7	4	12
P_{256}	256	8	4	12
P_{288}	288	6	8	12

使用 PHOTON 对一个消息进行杂凑，首先填充消息，使得填充之后的长度为每次吸收比特数 r 的倍数。而后将它们分为 r 比特的块 $M_1, \cdots, M_N$。每一次吸收，状态 S_i 吸收消息块 M_i，然后进入置换 P_t 更新状态。可以用下式表示。

$$S_{i+1} = P_t(S_i \oplus (M_i \| \{0\}^c))$$

所有消息块都被吸收之后，便开始榨取部分。摘要值就是将每次 r' 比特的输出连结起来，直到达到所需长度[2, 3]。

2.2　JH 杂凑函数

JH 是进入 SHA-3 竞赛最终轮的 5 个杂凑函数之一。它是一个迭代型的杂凑函数，其设计理念有 2 个重要部分：1）多维 AES 的设计；2）一个新的压缩函数结构。JH 的压缩函数采用一种新的结构，如图 2 所示。

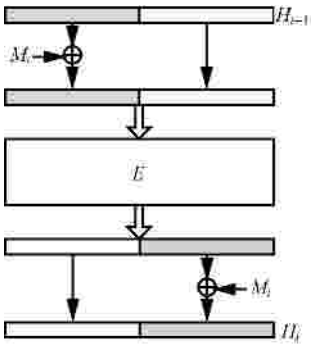


图 2　JH 的压缩函数结构

此压缩函数迭代地应用于 $H_1, H_2, \cdots, H_N$

$$H_i = F_d(H_{i-1}, M_i)$$
$$= E_d(H_{i-1} \oplus (M_i \| 0^{2^{d+1}})) \oplus (0^{2^{d+1}} \| M_i)$$

其中，F_d 为压缩函数，E_d 为一个双射函数。本文中称 E_d 为 JH 的核心函数。d 表示 JH 状态的维数，即其状态可以记为一个 d 维的矩阵。笔者推荐 d=8。

JH 杂凑函数处理的消息长度需要是 512bit 的

倍数，最后生成的摘要长度为 224bit、256bit、384bit 或 512bit。为了使处理的消息比特长度为 512 的倍数，需对消息按一定规则进行填充，填充后的消息被分为 N 个 512bit 的块，$M_1, M_2, \cdots, M_N$。

核心函数中，2^{d+2} 个输入首先用“比特切片”的方式分为 2^d 个 4bit 大小的“半字节”。而后，这些半字节经过包含 $6(d-1)$ 轮的轮函数 R_d。最后再应用开始的“比特切片”的逆变换。轮函数 R_d 共包含 3 种运算。

1) S 盒：JH 有 2 个 4×4 的 S 盒，每次使用哪一个取决于轮常数。

2) 线性扩散层：运算 L 将 2 个 4bit 的字(A,B)变为 2 个 4bit 字(C,D)，运算规则为

$$(C,D)=L(A,B)=(5A+2B,2A+B)$$

3) 置换 P_d：交换状态中“半字节”的次序，与 AES 轮函数中的行移位（shiftrows）运算相似。

JH 的轮函数 R_d 的一轮可以用 $P_d \circ L \circ S$ 来表示。以 d=4 时为例，设 $a_1, \cdots, a_{15}$ 和 $b_1, \cdots, b_{15}$ 分别为一轮的输入输出，则轮函数 R_4 的一轮可以用图 3 来表示。

笔者推荐的版本中维数为 8，所以核心函数的总轮数为 42。此外，状态大小为 1 024bit，生成摘要的时候截取最后一个链值 H_N 的一部分比特[4]。

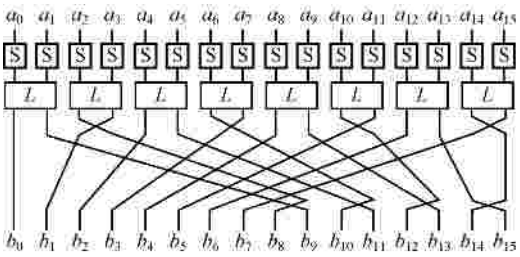


图 3 R_4 的一轮

3 高阶差分与零和区分器

3.1 高阶差分

1994 年，学者来学嘉给出了密码函数的微分的定义[9]。

定义 1 设 P 为一个 F_2^n 上的置换。对于任意的 $a \in F_2^n$，P 在点 a 处的微分为函数

$$D_a P(x)=P(x\oplus a)\oplus P(x)$$

而 P 在点 $a_1,\cdots,a_i$ 处的 i 阶微分定义为

$$D_{a_1,\cdots,a_i}P(x)=D_{a_i}(D_{a_{i-1}}(\cdots D_{a_1}P(x)))$$
$$=\bigoplus_{(k_1,\cdots,k_i)\in F_2^i} P(x\oplus\bigoplus_{j=1}^{i}k_j a_j)$$

对于任意的 F_2^n 上的 m 维子空间 V，P 在 V 上的 m 阶微分为函数

$$D_V P(x)=D_{a_m}(D_{a_{m-1}}(\cdots D_{a_1}P(x)))=\bigoplus_{v\in V}P(x\oplus v)$$

其中，$a_1, \cdots, a_m$ 为 V 的一组基。

一个函数的一阶微分的代数次数有性质 $\deg(D_a P(x)) \le \deg P(x)-1$。所以，对于任意的 $(\deg P+1)$ 维的子空间 V 或者是任意的 $(\deg P+1)$ 个点 $a_1, \cdots, a_{\deg P+1}$，置换 P 的 $(\deg P+1)$ 阶微分为 0。

3.2 积分攻击

积分攻击[15~18]是一种基于字节的分组密码攻击方法。这种攻击的基本思想是：考虑一些值和的传播性质，并利用这些性质来验证所猜测密钥。

攻击中，一些明文字节被选定为常数，记为 C，其他字节，一般称为活跃字节，取遍所有的 256 个值，这些字节记为 A。对于活跃字节来说，通过 S 盒变换之后仍然是活跃的。也就是说，活跃字节通过 S 盒之后的像仍然取遍所有的 256 个值。而且通过轮常数加运算之后，活跃性也不变。如果通过行移位运算，活跃字节的位置会发生变化。

文献[15,17]还提出了高阶积分的思想。和原始的积分攻击不同，高阶积分的起始点为多个活跃字节，这些字节之间独立，它们在通过开始的几轮 S 盒和线性扩散层之后仍然是活跃的。原因为 S 盒和线性扩散层都是置换，只要输入取遍所有情况，输出一定也可以取遍。当然，利用积分攻击也可以攻击基于“半字节”的算法。

3.3 零和区分器

零和区分器由学者 J P Aumasson 于 2009 年提出[11]。

定义 2 设 F 为一个 F_2^n 上的函数。如果 F 的输入集，$\{x_1, \cdots, x_K\} \subset F_2^n$，满足下面 2 个条件。

1) 所有这些元素的和为 0。

2) 所有这些元素在 F 下的像的和为 0。

则称其为 F 的大小为 K 的零和。

本文以高阶差分为工具，来构造某些密码函数的零和区分器。由前面的高阶差分的性质，如果已知一个 F_2^n 上的置换 P 的代数次数的上界 l。在 P 的输入中任意选择 $n-(l+1)$ 个比特位置，选定这些比特的值，遍历剩下的 $(l+1)$ 个比特（称其为活跃比特）的所有可以取到的值。则由高阶差分性质，这 2^{l+1} 个输入的输出之和为 0。

对于由轮函数迭代构成的置换，这里用 $P = R_r \circ \cdots \circ R_1$ 来表示，选择合适的整数 t，$1 \le t \le r$，将置换 P 分成 2 部分，$F_{r-t} = R_r \circ \cdots \circ R_{t+1}$ 和 $G_t = R_1^{-1} \circ \cdots \circ R_t^{-1}$。令

$$d = \max(\deg(F_{r-t}), \deg(G_t))$$

如果中间状态含有的比特数大于 d+1，可以在第 t 轮的输出状态中选择 d+1bit 作为活跃比特，向两边计算，最后在两端得到平衡的状态，即输入输出的和都为 0（如图 4 所示）。

图 4　零和攻击结构

如果 F 为 F_2^n 上的函数，则同样尺寸的随机函数的输出之和为 0 的概率为 2^{-n}。如果可以以较大的概率（本文中方法的概率为 1）找到 F 的零和，则可以说，就构造了函数 F 的一个零和区分器。

4　构造零和区分器的新方法

本节介绍一种针对 AES 类置换，构造零和区分器的新方法。这种方法组合高阶积分攻击和高阶差分攻击，针对 AES 类置换的扩散特点，构造一条由高阶积分路径和高阶差分路径组合而成的组合路径。

本文所研究的 AES 类置换，状态可以表示成一个方阵，轮函数使用的是 SP 结构，各单元需要平行独立地通过 S 盒，然后一列（或者一行）再进入一个线性扩散层进行混淆。具体来说，考虑 F 为一个 F_2^n 上的由轮函数迭代而成的 AES 类置换，其状态为一个 d 维方阵，状态中单元大小为 $s$bit，每行（或者每列）含有 b 个单元，即状态大小为 sb^dbit。轮函数使用的 SP 结构满足，状态先通过多个置换 S 盒并置的代换层，然后再进入一个一维的线性扩散层 P。除此之外还有一个交换单元位置的移位运算和一个常数加运算（如果没有常数加，可以看作加上全为 0 的常数）。设 P 的分支数达到最大的 b+1，则对于这类置换，有下面结果。

命题 1　置换 F 如上所述，则选定状态中的一个单元，达到完全扩散（即某中间状态中每个单元都与所选单元相关）至少需要 d 轮；同样地，逆向计算达到完全扩散也至少需要 d 轮。

证明　由于线性扩散层只在一维（即一行或者一列）上进行，而其分支数达到最大的 b+1，所以经过一轮运算，状态矩阵中有 b 个单元与所选单元相关。考虑到单元换位运算，两轮之后至多有 b^2 个单元与所选单元相关。由此继续推导，则经过 i 轮运算，至多有 b^i 个单元与所选单元相关。所以，由于状态为一个 d 维方阵，若使得所有 b^d 个单元与所选单元相关，至少需要 d 轮。逆向结果利用同样证明过程可得。证毕。

命题 2　置换 F 如上所述，设其 d 轮可达到完全扩散。选定状态中的一个单元，则从此状态出发，第 d–1 轮的输出状态中有 b^{d-1} 个单元与所选单元相关，即第 d 轮的 S 盒计算中有 $1/b$ 与其相关；同样地，逆向计算第 d–1 轮的输出状态中也有 b^{d-1} 个单元与所选单元相关，但逆向第 d 轮的 S 盒全部与其相关。

证明　由于使用的一维线性扩散层，而达到完全扩散需要 d 轮，所以 d–1 轮之后能且只能有 b^{d-1} 个单元与所选单元相关。再进入第 d 轮的 S 盒层时有 b^{d-1} 个 S 盒计算与其相关。而对于逆向计算，易知同样有 d–1 轮之后能且只能有 b^{d-1} 个单元与所选单元相关。但是，由于逆向计算第 d 轮时，先进入线性扩散层 P，然后进入 S 盒层，所以逆向第 d 轮的 S 盒全部与其相关。证毕。

命题 3　置换 F 如上所述，并且 d 轮可达到完全扩散。则至多需要选择 $b^d - 1$ 个单元作为高阶积分攻击中的活跃单元，即取遍这些单元中的所有 $2^{s(b^d-1)}$ 种情况，剩余的一个单元取定值，记为 C_i，能够使得：

1) 正向计算，第 d 轮的 S 盒层的输出中，有 $(b-1)b^{d-1}$ 个单元是活跃且独立的（相互独立，并且与其他 b^{d-1} 个单元也无关，下文中的“独立”也代表同一意思）；

2) 逆向计算，第 d–1 轮的 S 盒层的输出中，有 $(b-1)b^{d-1}$ 个单元是活跃且独立的。

证明　由命题 2 可知，正向计算第 d–1 轮，有 b^{d-1} 个单元与取定值的单元 C_i 相关。又由于 S 盒层和一维线性扩散层，包括单元换位运算和常数加运算，都是置换，即双射，所以其他的 $(b-1)b^{d-1}$ 个单元是活跃且独立的。这些单元在经过第 d 轮的 S 盒层后，仍然是活跃且独立的。

而对于逆向计算，由于 S 盒层在扩散层 P 之后，所以只能保证在第 d–1 轮的 S 盒层的输出中，有 $(b-1)b^{d-1}$ 个单元是活跃且独立的。证毕。

对于某个具体算法，可能不必选择 $b^d - 1$ 个活跃单元，有时更少的活跃单元也可以得到同样的结

果。以已知密钥的 AES 为例，取 15 个字节为活跃字节，剩下的一个字节取定值，则通过第 2 轮的 S 盒层之后，仍然有 12 个字节活跃且独立。但是事实上，取状态矩阵中主对角线之外的 12 个字节为活跃字节，也能达到这个结果。

基于这些结果，针对上述 d 轮可完全扩散的 AES 类置换，可以利用高阶积分攻击和高阶差分攻击思想，用如下步骤构造零和区分器。

1) 选择一个中间状态，称其为起始状态，在此状态中选择一个单元。对此单元任选一个定值，而对其他单元取遍所有的 $2^{s(b^d-1)}$ 种情况，也就是说，其他单元为高阶积分攻击中的活跃单元。

2) 由命题 3，正向计算第 d 轮的 S 盒层之后有 $(b-1)b^{d-1}$ 个单元是活跃且独立的。另一方面，可以保证在逆向计算 d-1 轮后，有 $(b-1)b^{d-1}$ 个单元是活跃且独立的。这样就建立了一条$(2d-1.5)$轮的高阶积分路径。

3) 计算若干轮正向迭代后的代数次数。如果 j 轮迭代后的代数次数小于 $s(b-1)b^{d-1}$，而 j+1 轮代数次数大于等于 $s(b-1)b^{d-1}$，则以上述高阶积分路径的终点作为起始点，进行 j 轮的高阶差分攻击。虽然高阶差分路径是从第 2)步剩下的一个 P 置换（0.5 轮）开始，但是由于 P 置换是线性的，也就是说代数次数为 1，不影响高阶差分路径的代数次数，所以在高阶差分路径的终点得到平衡状态。逆向的攻击和正向相似。

这样就得到了一个 $(2d-1+j+j')$ 轮的零和区分器。图 5 表示了整个攻击过程。

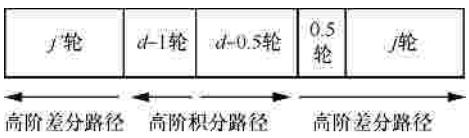


图 5 改进的零和攻击结构

值得注意的是，针对某些具体算法，有时可以选取具有更少活跃单元的中间起始状态，来降低攻击的复杂度。但是，有些情况更少的活跃单元可能会导致攻击的轮数略有减少。

5 2 个 PHOTON 置换的改进零和攻击

PHOTON 杂凑函数族中采用的置换为 d=2 维 AES 结构，S 盒层采用的是 PRESENT 的 4bit S 盒，或者是 AES 的 8bit S 盒，所以 s=4 或 s=8。状态中单元为字节或半字节，个数 b^2 分别为 25，36，49，64 和 36。扩散层分支数均达到最大，并且经过 2 轮可以达到完全扩散。则在中间可以构造一条 1+1.5=2.5 轮的高阶积分路径，并且在积分路径的两端保证有 $sd(d-1)$个比特活跃。这样可以对置换 P_{196} 和 P_{256} 进行 13 轮的零和攻击。但是，这 2 个置换全轮只有 12 轮，基于这个考虑，如果选择合适的单元作为活跃单元，可以在攻击轮数减少一轮的情况下，大大降低攻击的复杂度。

5.1 对 P_{196} 的改进零和攻击

P_{196} 采用 4bit S 盒，状态中含有 7^2=49 个“半字节”。攻击中每一步攻击的轮数为

$$5 \xleftarrow{3)} 1 \xleftarrow{2)} \xrightarrow{2)} 0.5 \xrightarrow{3)} 5.5$$

具体攻击过程如下。

1) 在整个 12 轮置换中，选择第 6 轮的输出作为起始状态，并选择状态中的前 6 列活跃，即选择最后 1 列为定值，遍历前 6 列的 2^{168} 种情况（如图 6 所示，灰色表示活跃半字节，白色表示非活跃半字节，取图中 MC 的输出，即第 6 轮的输出为起始状态）。

2) 正向计算，这 6 个活跃列在通过第 7 轮的 S 盒之后仍然是活跃且独立的，后面的行移位和线性扩散层归到高阶差分路径中去，并不影响其代数次数；而逆向计算，由于列混淆运算是一个置换，所以，这 6 列在通过第 6 轮的列混淆的逆之后仍然活跃且独立，通过行移位的逆之后活跃半字节只是位置发生变化，通过 S 盒层的逆之后仍然有 42 个活跃且独立的半字节，包含 168bit。

3) 由 5 轮代数次数至多为 157，并且其逆函数次数与其相同可知，从 168 个活跃比特出发，经过 5 轮之后的状态是平衡的。

这样就以 2^{168} 的复杂度，构造了一个 P_{196} 的零和区分器。

5.2 对 P_{256} 的改进零和攻击

P_{256} 采用 4bit S 盒，状态中含有 8^2=64 个“半字节”。每一步攻击的轮数与上述对 P_{196} 的攻击中的轮数相同。攻击过程也是从选择第 6 轮输出开始，选择前 7 列活跃，则正向计算 0.5 轮，逆向计算 1 轮之后仍然有 224 个活跃比特，而其 5 轮的代数次数至多为 197，所以可以在高阶差分路径的两端得到平衡状态。

这样就以 2^{224} 的复杂度，构造了一个 P_{256} 的零

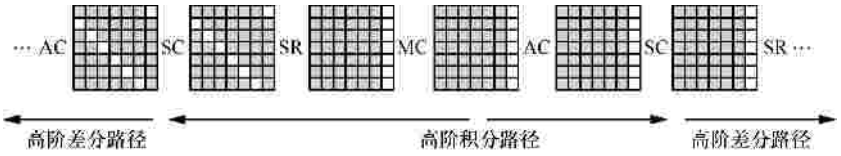


图 6　P_{196} 零和攻击的中间 1.5 轮

和区分器。

6　JH 核心函数的零和区分器

本节对 JH 的核心函数进行零和攻击。首先分析了一下 4 维的核心函数，然后在此基础上得到 8 维的分析结果。

JH 核心函数的轮函数是一个广义的 AES 构造。它以 4bit 的“半字节”为单元，先通过一个 S 盒层提供非线性性，然后是 4bit 元素进行两两线性混淆，最后是一个元素换位。这里的 S 盒是 4bit 入、4bit 出的平衡 S 盒。线性扩散层是一个双射，分支数为 2。考虑积分攻击，2 个活跃且独立的“半字节”通过此扩散层，仍然是 2 个活跃且独立的“半字节”。如果一个活跃，一个非活跃或者 2 个活跃不独立的“半字节”通过此扩散层，则输出 2 个“半字节”不能达到活跃且独立的要求。

这里，参数 b=2，s=4，对于一个 d 维状态来说，共有 2^d 个“半字节”，状态大小为 2^{d+2}bit，并且正向和逆向达到完全扩散均需要 d 轮，和状态的维数相等。

6.1　4 维 JH 核心函数 E_4 的零和攻击

对 4 维 JH 核心函数 E_4，可以构造一个 15 轮的零和区分器（总轮数为 18）。攻击同样包含 3 个步骤，每一步攻击的轮数为

$$4 \xleftarrow{3)} 3 \xleftarrow{2)} \xrightarrow{2)} 3.5 \xrightarrow{3)} 4.5$$

具体攻击过程如下。

1) 根据第 4 节的方法，首先需要在第 7 轮的输出状态中选定一个“半字节”的值，遍历其他 15 个“半字节”的值。这样在第 10 轮中 S 盒层的输出状态和第 4 轮的输出状态中，存在 $(2-1)2^{4-1}=8$ 个活跃且独立的“半字节”。但是，由 JH 轮函数的扩散特点，如果选择后 8 个“半字节”作为活跃“半字节”，正向计算，在第 10 轮 S 盒层输出的第 2、4、6、8、9、11、13、15 个“半字节”活跃且独立，个数仍然是 8 个；并且，如果选择第 2、4、6、8、9、11、13、15 个“半字节”作为活跃“半字节”，逆向计算 3 轮，在第 4 轮的输出状态中，后 8 个“半字节”活跃且独立。综合正向逆向考虑，在第 7 轮的输出状态中，只需选择后 8 个“半字节”和前 8 个“半字节”中的第 2、4、6、8 个作为活跃“半字节”，即可保证在第 4 轮的输出状态和第 10 轮的 S 盒层输出中存在 8 个活跃且独立的“半字节”，共包含 32 个比特。如图 7 所示，活跃且独立“半字节”用加粗的黑线表示。需要注意的是，只有 2 个粗黑线入的扩散层 L 才能保证 2 个粗黑线出，即满足活跃且独立的条件，其他情况均不行。

2) 正向计算 3.5 轮，逆向计算 3 轮，在第 10 轮中 S 盒层的输出和第 4 轮的输出，得到含有 8 个活跃且独立的“半字节”状态，共包含 32bit。需要注意的是，剩下的 8 个“非活跃半字节”的值并不是固定的。总的来说，它们也有 $2^{48-32}=2^{16}$ 种情况，但是，由于它们和 8 个“活跃半字节”是独立的，所以，对于每一种情况，8 个“活跃半字节”都会取遍所有的 2^{32} 种情况。如果每一种情况都会得到平衡状态，即和为 0 的状态，则综合起来考虑，和仍然为 0，仍然为平衡状态。

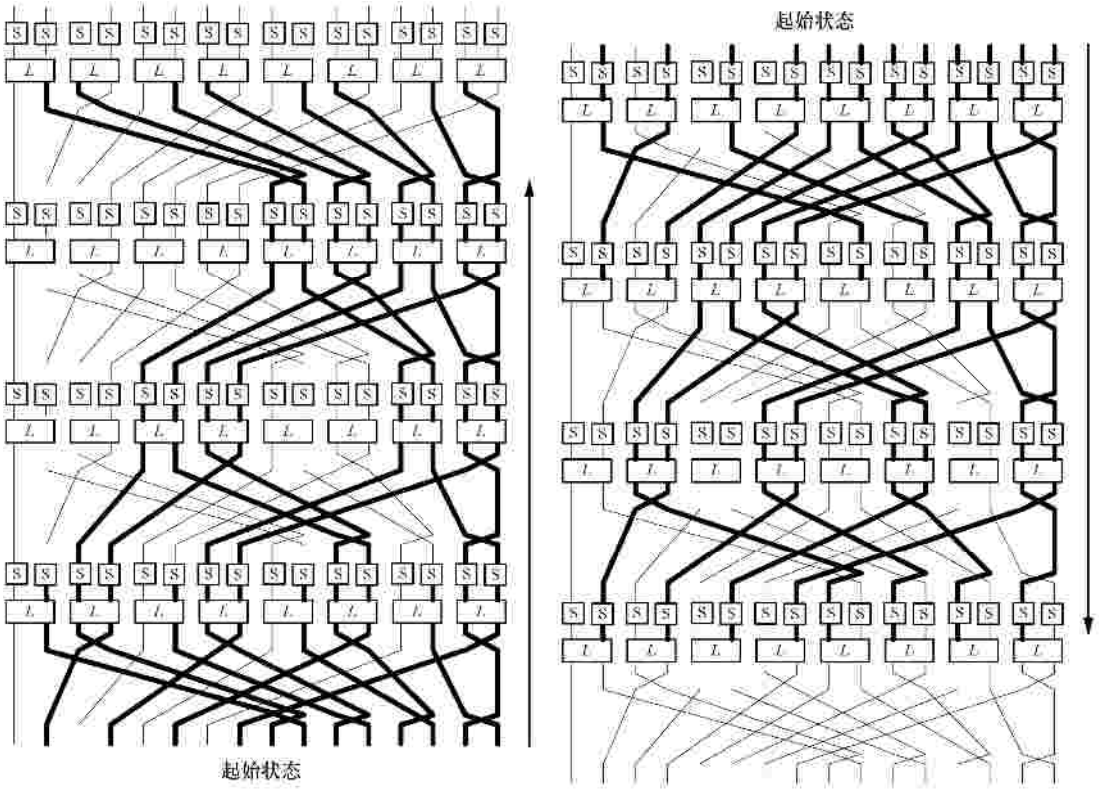
3) 由文献[8]中 Boura C 等给出的结果，4 轮的 JH 的代数次数至多是 25，所以加上次数为 1 的半轮，再正向计算 4.5 轮；此外，还可以再逆向计算 4 轮。

综上所述，以 2^{48} 的复杂度（事实上，只需选择全部“半字节”的 3/4 作为活跃“半字节”即可），对轮函数 R_4 构造一个 15 轮的零和区分器。事实上，由于逆向计算先进入一个线性扩散层 L，再进入 S 盒层，所以可以在最左端再加上 0.5 轮，得到和原来 4 轮逆向函数的代数次数相同的 4.5 轮逆向函数，使得攻击复杂度不变，而攻击轮数增加 0.5 轮。

和轮函数 R_4 相比，核心函数 E_4 多了开始的“比特切片”和最后的“比特切片”逆运算。这只会变换比特的位置，不影响其平衡性。至此，得到了核心函数 E_4 的 15.5 轮的零和区分器。

6.2　8 维 JH 核心函数 E_8 的零和攻击

JH 的推荐版本中，维数 d=8。参照 4 维版本的攻击过程对其进行攻击，构造一个 31 轮的零和区分器（总轮数为 42），每一步的攻击轮数为



(a) 逆向3轮高阶积分路径 (b) 正向4轮高阶积分路径

图 7 P_4 构造的 7 轮高阶积分路径

$$8 \xleftarrow{3)} 7 \xleftarrow{2)} \xrightarrow{2)} 7.5 \xrightarrow{3)} 8.5$$

根据 4 维版本的攻击步骤，首先需要在第 15 轮的输出状态中，选择$\frac{3}{4}\times 2^8 = 2^{192}$个活跃“半字节”，其他的取为定值。分别进行正向和逆向计算，在第 23 轮中 S 盒层的输出状态,和第 8 轮的输出状态中，有$(2-1)\times 2^{8-1} = 128$个活跃且独立的“半字节”，共包含 512bit。这样就可以以积分路径的终点作为起始点，来构造高阶差分路径。8 轮的 JH 核心函数的代数次数至多是 409[8]，所以用具有 512 个活跃比特的状态作为起始点，8 轮之后必为平衡状态。加上次数为 1 的半轮，可以正向计算 8.5 轮；此外，还可以再逆向计算 8 轮。

这样以$2^{\frac{3}{4}\times 2^{8+2}} = 2^{768}$的复杂度，对 R_8 构造一个 31 轮的零和区分器。同样地，考虑一轮中的运算的顺序和“比特切片”，得到了核心函数 E_8 的 31.5 轮的零和区分器。

而根据零和区分器特点，在构造的时候只需要找到其所需的输入即可，也就是说，只需进行逆向计算，这样可以节省大约一半的时间，所以其时间复杂度为 2^{767}。

在 SHA-3 竞赛第 1、第 2 轮中，JH 采用的轮数为 5(d-1)+0.5，对其轻量级版本，即 d=4 时的核心函数来说，其轮数为 15.5，而本文中构造的零和区分器也可以区分 4.5+3+4+4=15.5 轮。基于安全性考量，本文在第 3 轮将轮数增加到 6(d- 1)。

7 结束语

本文根据 AES 类置换的特点,提出了一种针对这类置换构造零和区分器的新方法。这种方法是高阶积分攻击和高阶差分攻击的组合。如果有若干轮轮函数代数次数上界的一个估计，则可以根据此上界，在中间先构造一条高阶积分路径，然后以这条高阶积分路径的 2 个终点作为起始点，再进行高阶差分攻击。

利用这一方法，对 PHOTON 算法中的 2 个置换进行零和攻击，分别以 2^{168} 和 2^{224} 的复杂度，构造了 P_{196} 和 P_{256} 的零和区分器。此外，还对 JH 中的置换进行零和攻击，以 2^{768} 的复杂度构造了 JH 核心函数的 31.5 轮零和区分器。如何构造出轮数更多的区分器值得将来的研究和关注。

参考文献：

[1] JOAN D, VINCENT R. The Design of Rijndael: AES—the Advanced Encryption Standard[M]. Springer-Verlag, 2002.

[2] GUO J, THOMAS P, AXEL P. The photon family of lightweight hash functions[EB/OL]. http://eprint.iacr.org/2011/609.pdf, 2011.

[3] GUO J, THOMAS P, AXEL P. The Photon family of lightweight hash functions[A]. Proc of the Advances in Cryptology-CAYPTO 2011[C]. Santa Barbara, CA, USA, 2011. 222-239.

[4] WU H. The Hash Function JH[EB/OL]. http://icsd.i2r.a-star.edu.sg/staff/hongjun/jh/jh_round2.pdf, 2008.

[5] RIJMEN V, TOZ D, VANCI. Rebound attack on reduced-round versions of JH[A]. Proc of the Fast Software Encryption-FSE 2010[C]. Seoul, Korea, 2010. 286-303.

[6] TURAN M-S, UYAN E. Practical near–collisions for reduced round blake, fugue, Hamsi and JH[EB/OL]. http://csrc.nist.gov/ groups/ST/hash/sha-3/Round2/Aug2010/documents/papers/TURAN_Paper_Erdener.pdf, 2010.

[7] N-PLASENCIA M. How to improve rebound attacks[A]. Proc of the Advances in Cryptology-CAYPTO 2011[C]. Santa Barbara, CA, USA, 2011. 188-205.

[8] BOURA C, CANTEAUT A. On the influence of the algebraic degree of F-1 on the algebraic degree of G circ F[EB/OL]. http://eprint.iacr.org/ 2011/503.pdf, 2011.

[9] LAI X J. Higher order derivatives and differential cryptanalysis[A]. Proc of Symposium on Communication, Coding and Cryptography, in honor o] James L. Massey on the Occasion of His 60'th Birthday[C]. Monte-Verita, Ascona, Switzerland, 1994. 10-13.

[10] KNUDSEN L R. Truncated and higher order differentials[A]. Proc of the Fast Software Encryption-FSE 1994[C]. Leuven, Belgium, 1995. 196-211.

[11] AUMASSON J-P, MEIER W. Zero-sum distinguishers for reduced keccak-f and for the core functions of luffa and hamsi[EB/OL]. http://www.131002.net/data/papers/AM09.pdf, 2009.

[12] BOURA C, CANTEAUT A. Zero-sum distinguishers for iterated permutations and application to KECCAK-f and Hamsi-256[A]. Proc of the Selected Areas in Cryptography-SAC 2010[C]. Waterloo, Ontario, Canada, 2011. 1-17.

[13] BOURA C, CANTEAUT A, CANNIÈRE C D. Higher-order differential properties of Keccak and Luffa[A]. Proc of the Fast Software Encryption-FSE 2011[C]. Lyngby, Denmark, 2011. 252-269.

[14] AUMASSON J-P, KÄSPER E, KNUDSEN L R, *et al*. Distinguishers for the compression function and output transformation of hamsi-256[A]. Proc of the Information Security and Privacy-ACISP 2010[C]. Sydney, Australia, 2010. 87-103.

[15] KNUDSEN L R, WAGNER D. Integral cryptanalysis[A]. Proc of the Fast Software Encryption-FSE 2002[C]. Leuven, Belgium, 2002. 112-127.

[16] DAEMEN J, KNUDSEN L R, RIJMEN V. The block cipher square[A]. Proc of the Fast Software Encryption-FSE 1997[C]. Haifa, Israel, 1997. 149-165.

[17] FERGUSON N, KELSEY J, LUCKS S, *et al*. Improved cryptanalysis of rijndael[A]. Proc of the Fast Software Encryption-FSE 2000[C]. New York, NY, USA, 2001. 213-230.

[18] GALICE S, MINIER M. Improving integral attacks against rijndael-256 up to 9 rounds[A]. Proc of the Progress in Cryptology – AFRICACRYPT 2008[C]. Casablanca, Morocco, 2008. 1-15.

作者简介：

董乐（1980-），男，河南新乡人，中国科学院软件研究所博士生，主要研究方向为杂凑函数和分组密码的分析。

吴文玲（1966-），女，陕西蒲城人，博士，中国科学院软件研究所研究员、博士生导师，主要研究方向为私钥密码体制的设计与分析。

吴双（1983-），男，重庆人，博士，中国科学院软件研究所助理研究员，主要研究方向为杂凑函数的分析与设计。

邹剑（1985-），男，福建福州人，中国科学院软件研究所博士生，主要研究方向为杂凑函数分析。