改进的减轮 Kiasu-BC 算法的中间相遇攻击

李曼曼[1,2,3]，陈少真[1,2,3]

（1. 信息工程大学网络空间安全学院，河南 郑州 450001；2. 密码科学技术国家重点实验室，北京 100878；
3. 河南省网络密码技术重点实验室，河南 郑州 450001）

摘 要：Kiasu-BC 算法是加密认证竞赛 CAESAR 第一轮入选方案 Kiasu 的内置可调分组密码。Kiasu-BC 算法是基于 AES-128 轮函数构造的可调分组密码算法，通过对 Kiasu-BC 算法的结构特征进行研究，利用调柄自由度以及内部密钥间的制约关系，降低预计算的复杂度。结合差分枚举技术，构造新的 5 轮中间相遇区分器，改进 Kiasu-BC 算法的 8 轮中间相遇攻击。改进后攻击的时间复杂度为 2^{114}，存储复杂度为 2^{63}，数据复杂度为 2^{108}。

关键词：可调分组密码；Kiasu-BC 算法；中间相遇攻击；差分枚举

中图分类号：TN918.1

文献标志码：A

DOI: 10.11959/j.issn.1000−436x.2022112

Improved meet-in-the-middle attack on reduced-round Kiasu-BC algorithm

LI Manman[1,2,3], CHEN Shaozhen[1,2,3]

1. College of Cyberspace Security, Information Engineering University, Zhengzhou 450001, China
2. State Key Laboratory of Cryptology, Beijing 100878, China
3. Henan Key Laboratory of Network Cryptography Technology, Zhengzhou 450001, China

Abstract: Kiasu-BC algorithm is an internal tweakable block cipher of authenticated encryption algorithm Kiasu as one of first-round candidates in the CAESAR competition. The precomputation complexity is reduced by utilizing the freedom of the tweak and the internal key restriction through the research on structural characteristics of Kiasu-BC algorithm based on AES-128 round function. Combined with the differential enumeration technique, a new 5-round meet-in-the-middle distinguisher was constructed to improve the meet-in-the-middle attack on 8-round Kiasu-BC algorithm. The improved attack requires the time complexity of 2^{114}, the memory complexity of 2^{63} and the data complexity of 2^{108}.

Keywords: tweakable block cipher, Kiasu-BC algorithm, meet-in-the-middle attack, differential enumeration

0 引言

随着高新技术的飞速发展，以及网络支付系统、云计算、物联网技术的日益成熟，人们对隐私保护及信息安全等相关问题越来越重视。分组密码由于其软硬件实现效率高、易于标准化等优点被广泛用于保护信息的安全。可调分组密码是一种带有额外输入调柄值的分组密码，调柄值可以提高算法的灵活性，在构造 Hash 函数、消息认证码和伪随机数发生器等领域应用广泛。近些年，可调分组密码迅速发展，其设计方法与安全性分析得到人们诸多关注和研究。在 CRYPTO 2002 上，Liskov 等[1]提出了可调分组密码的概念。与传统的分组密码相比，可调分组密码有一个额外的输入调柄，这是一个完全公开的参数，增加了分组密码的可变性。在实际使用过程中，更换调柄比更换密钥操作更便捷

收稿日期：2022−02−09；**修回日期：**2022−05−09

基金项目：河南省网络密码技术重点实验室开放课题基金资助项目（No.LNCT2019-S03）

Foundation Item: Henan Key Laboratory of Network Cryptography Technology Open Funds (No.LNCT2019-S03)

有效且成本低廉。因此，可调分组密码被广泛应用于加密协议、认证加密等场合，以保护数据的机密性和认证性。

Jean 等[2]在 ASIACRYPT 2014 上提出了一个可用来设计可调分组密码的简单框架——可调密钥（Tweakey）模型，并给出了 3 个基于 AES 算法轮函数的可调分组密码的具体实例，分别是 Kiasu-BC[3]、Joltik-BC[4]和 Deoxys-BC[5]。近几年，针对 Kiasu-BC 算法的安全性分析，研究者使用多种密码分析方法给出了较好的分析结果。最初，Kiasu-BC 算法的设计者分析了 Kiasu-BC 算法的中间相遇攻击和差分攻击，并且声称 Kiasu-BC 算法与 AES-128 算法持有相同的轮函数和密钥扩展算法，故其抵抗密码分析方法的能力与 AES-128 算法相同。

之后，研究者对 Kiasu-BC 算法安全性分析的主要结果如下。Dobraunig 等[6]利用调柄自由度构造了 4 轮区分器，实现了 7 轮 Kiasu-BC 算法的积分攻击；Abdelkhalek 等[7]利用调柄生成的差分可抵消攻击路径的差分首次构造了 8 轮 Kiasu-BC 算法的不可能差分攻击；Tolba 等[8]利用调柄差分增加区分器轮数实现了 8 轮 Kiasu-BC 算法的中间相遇攻击；Dobraunig 等[9]对 8 轮 Kiasu-BC 算法的不可能差分攻击进行了改进，并提出了 8 轮 Kiasu-BC 算法的飞去来器攻击；Jiang 等[10]借鉴调柄生成的非零差分抵消攻击路径差分的思想，提出了 8 轮 Kiasu-BC 算法的多重不可能差分攻击。

本文主要研究 Kiasu-BC 算法的中间相遇攻击。Kiasu-BC 算法的中间相遇攻击最初由 Jean 等[3]提出。Tolba 等[8]利用 Kiasu-BC 算法的调柄差分性质，通过控制调柄差分使区分器在其第一轮处差分为 0，据此将区分器的轮数扩展到 5 轮，在构造区分器的前面增加一轮、后面增加 2 轮，实现了 8 轮中间相遇攻击。Liu 等[11]结合差分枚举技术选取不同于文献[9]的活动字节对中间相遇区分器进行了改进，更新了 8 轮中间相遇攻击的结果，降低了 Kiasu-BC 算法中间相遇攻击的时间复杂度和数据复杂度。关于 Kiasu-BC 算法的中间相遇攻击，本文在前人研究的基础上，结合 Kiasu-BC 算法的性质，寻找其密钥扩展算法的特点，发现 Kiasu-BC 算法内部密钥间的关联性，利用 Kiasu-BC 算法密钥扩展的缺陷和轮变换的特点，构造一个新的 5 轮中间相遇区分器，然后利用该区分器，向前扩展一轮、向后扩展 2 轮，改进了 8 轮 Kiasu-BC 算法中间相遇攻击的结果。改进后的 8 轮 Kiasu-BC 算法中间相遇攻击的存储复杂度为 2^{63}，时间复杂度为 2^{114}，数据复杂度为 2^{108}。

1 预备知识

1.1 符号说明

本文用到的符号说明如表 1 所示。

表 1 符号说明

符号	说明
m，n	状态矩阵的行数，列数
P，C	明文，密文
ΔP，ΔC	明文差分，密文差分
x_i，y_i，z_i，w_i	第 i 轮中字节代替、行移位、列混合、轮调柄密钥加变换前的状态
$\boldsymbol{x}[i]$	状态矩阵 $\boldsymbol{x}$ 的第 i 单元
$\boldsymbol{x}[i,\cdots,j]$	状态矩阵 $\boldsymbol{x}$ 的第 i 至第 j 单元
$x\|\|y$	x 和 y 级联
$x^{\lll k}$	状态 x 循环左移 k bit

1.2 Kiasu-BC 算法介绍

Kiasu-BC 算法是在 AES-128 算法的基础上设计的可调分组密码算法，采用代换−置换网络（SPN, substitution-permutation network）结构，明文分组长度和密钥长度均为 128 bit，加密轮数为 10 轮。Kiasu-BC 算法的结构如图 1 所示。Kiasu-BC 算法中调柄 T 的结构如图 2 所示。

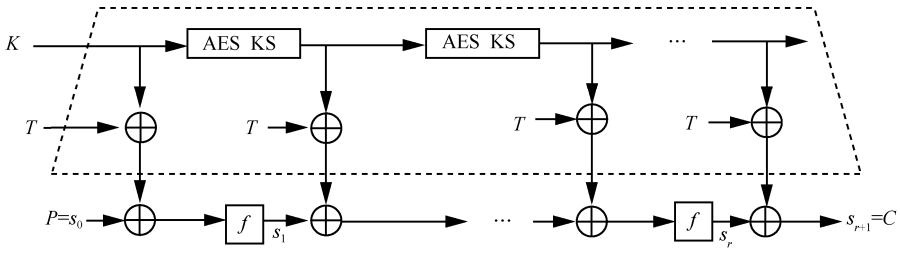

图 1 Kiasu-BC 算法的结构

$$T=\begin{array}{|c|c|c|c|}\hline T_0 & T_2 & T_4 & T_6\\ \hline T_1 & T_3 & T_5 & T_7\\ \hline 0 & 0 & 0 & 0\\ \hline 0 & 0 & 0 & 0\\ \hline\end{array}$$

图 2　调柄 T 的结构

Kiasu-BC 算法的 128 bit 明密文及中间状态可分成 16 块，表示为如下的 4×4 字节矩阵，每个字节可以看作 GF(2^8) 中的一个元素。

$$\begin{pmatrix} a_0 & a_4 & a_8 & a_{12}\\ a_1 & a_5 & a_9 & a_{13}\\ a_2 & a_6 & a_{10} & a_{14}\\ a_3 & a_7 & a_{11} & a_{15}\end{pmatrix}$$

Kiasu-BC 算法的轮函数与 AES-128 算法相似，分别由字节替换（SB, SubByte）、行位移（SR, ShiftRow）、列混合（MC, MixColumn）和轮调柄密钥加（ART, AddRoundTweakey）4 种变换构成，其中前 3 种变换与 AES-128 算法相同，ART 变换就是将 192 bit 轮调柄密钥 STK_i 与 128 bit 中间状态进行按位异或运算。其中，192 bit 轮调柄密钥由 128 bit 轮密钥 RK_i 和 64 bit 调柄 T 级联而成。Kiasu-BC 算法的轮调柄密钥生成如图 3 所示。

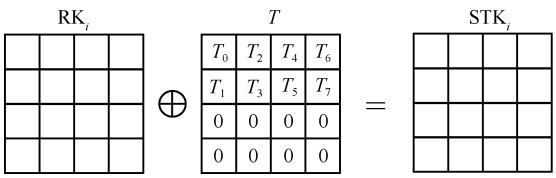

图 3　Kiasu-BC 算法的轮调柄密钥生成

另外，在 Kiasu-BC 算法的 128 bit 明文进入轮函数之前，需要先进行一个白化的轮调柄密钥加变换，并且最后一轮的列混合变换被省略。

Kiasu-BC 算法 128 bit 轮密钥 RK_i 是由 AES-128 的密钥扩展算法得到的，对初始密钥进行扩展并将其分配到加密过程的每一轮。

2　中间相遇攻击

中间相遇攻击是一种有效的选择明文攻击，由 Diffie 和 Hellman[12]于 1977 年分析 DES 算法时提出，该攻击方法利用密码分割和时空折中 2 种分析思想，在分组密码和 Hash 函数的安全性分析中均有广泛应用。随着 20 世纪 90 年代差分攻击的出现，许多密码分析方法逐渐融合形成新的、有效的分析思想。例如，不可能差分攻击就是利用截断差分在中间相遇产生矛盾的思想提出的，其本质也是中间相遇攻击。目前，研究者利用中间相遇攻击对许多分组密码算法进行了有效分析。在过去几年里，这种攻击方法被改进成一种名为原像攻击的方法，然后应用于 Hash 函数的攻击中，并据此提出许多新的技术。

2008 年，Demirci 和 Selcuk[13]在分析 AES 算法安全性的过程中，将碰撞攻击的思想[14]与中间相遇攻击相结合，提出了一种新型的中间相遇攻击模型，称为 Demirci-Selcuk 中间相遇攻击。其基本思想是将一个分组密码算法 E 分割成连续的 3 个部分，即 $E=E_2\circ E_0\circ E_1$，如图 4 所示。

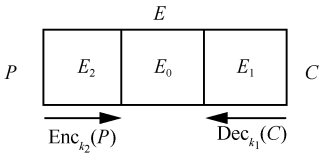

图 4　Demirci-Selcuk 中间相遇攻击的基本思想

攻击过程包含离线部分和在线部分。在离线部分 E_0 处，构造一个中间相遇区分器，用一个预计算表存储区分器中特定的输入和输出；在密钥恢复阶段即在线部分，猜测 E_1、E_2 的某些相关子密钥 k_1、k_2，加密选择明文，同时解密对应的密文，检测得到的值与预计算表中存储的值是否相匹配。如果匹配，那么猜测密钥是正确的；如果不匹配，那么猜测密钥是错误的，因此排除这个错误的猜测密钥。

Dunkelman、Keller 和 Shamir[15]在 2010 年提出了差分枚举技术，用来解决 Demirci-Selcuk 中间相遇攻击存储复杂度较大的问题，同时提出密钥桥技术，通过寻找密钥间的制约关系，降低攻击的时间复杂度。在 2010 年亚密会分析 AES 算法时，Dunkelman、Keller 和 Shamir 在 Demirci 思想的基础上引入多重集并利用有效的差分枚举手段，大大降低了中间相遇攻击预计算阶段的复杂度。2013 年，Derbez 等[16]利用反射攻击的思想对差分枚举技术进行了改进，利用改进后的差分枚举技术可以进一步降低 Demirci-Selcuk 中间相遇攻击的存储复杂度。

Li 等[17]在 2015 年提出相关密钥筛选技术，结合差分枚举技术可以进一步控制内部状态的取值

范围，降低 Demirci-Selcuk 中间相遇攻击的存储复杂度。随后，Demirci-Selcuk 中间相遇攻击技术被用于分析各种类型的分组密码算法[18-21]。

在 CRYPTO 2016 上，Derbez 和 Fouque[22]总结梳理了 Demirci-Selcuk 中间相遇攻击的攻击模式，利用计算机语言将 Demirci-Selcuk 中间相遇攻击的攻击流程进行转化，实现了 Demirci-Selcuk 中间相遇攻击的自动化搜索。借助计算机强大的计算能力，搜索最优的 Demirci-Selcuk 中间相遇攻击方案。

上述关于中间相遇攻击的改进技巧或方法为人们今后的研究提供了理论支撑，利用差分枚举、密钥桥等技术可以优化攻击结果。中间相遇攻击的自动化研究也是目前的一个研究热点，针对具体的算法给出自动化搜索的结果，这也是本文将要深入研究的一个问题。

3 Kiasu-BC 算法的 5 轮中间相遇区分器

本节进一步研究密钥扩展算法存在的缺陷，挖掘轮密钥间的相互制约关系，构造了一个新的 Kiasu-BC 算法 5 轮中间相遇区分器。在分析过程中，本文明确区分轮密钥和调柄，第 i 轮的轮密钥记作 k_i^0，第 i 轮的调柄记作 k_i^1。

3.1 Kiasu-BC 算法的性质

从密钥扩展算法可知，Kiasu-BC 算法每轮的最后一个变换所需要的轮密钥是由初始密钥经过扩展得到的。从上面描述的密钥扩展方案，本文可以得到下面的关系。

如果知道第 4 轮的轮密钥 k_4^0，即 $W[16],W[17],W[18],W[19]$，那么根据密钥扩展算法就可以得到如下关系

$$W[20]=W[16]\oplus \mathrm{SB}(W[19]^{<<8})\oplus \mathrm{RCON}[5]$$
$$W[21]=W[17]\oplus W[20]$$
$$W[22]=W[18]\oplus W[21]$$
$$W[23]=W[19]\oplus W[22]$$

用 $k_4^0[0,1,2,3],k_4^0[4,5,6,7],k_4^0[8,9,10,11],k_4^0[12,13,14,15]$ 表示 $W[16],W[17],W[18],W[19]$，则上面的关系可以表示为

$$k_5^0[0,1,2,3]=k_4^0[0,1,2,3]\oplus \mathrm{SB}(k_4^0[12,13,15,15]^{<<8})\oplus \mathrm{RCON}[5]$$
$$k_5^0[4,5,6,7]=k_4^0[4,5,6,7]\oplus k_5^0[0,1,2,3]$$
$$k_5^0[8,9,10,11]=k_4^0[8,9,10,11]\oplus k_5^0[4,5,6,7]$$
$$k_5^0[12,13,14,15]=k_4^0[12,13,14,15]\oplus k_5^0[8,9,10,11]$$

在攻击过程中，本文还会用到下列定义和性质。

定义 1[23] 算法的 b-δ-集是由其 2^b 个中间状态构成的集合，且这 2^b 个中间状态在某一个字节（活跃字节）处遍历，在其余字节（不活跃字节）处固定。

性质 1[24] 给定任意一个双射 S 盒 S，Δ_i 和 Δ_o 分别为随机选取的非零输入差分和非零输出差分，则方程 $S(x)\oplus S(x\oplus\Delta_i)=\Delta_o$ 平均只有一个解。

3.2 构造 5 轮中间相遇区分器

本节考虑 $w_0[0]$ 是活跃字节，经过 5 轮 Kiasu-BC 算法加密后，可以计算出相应的有序序列 $\Delta x_6[0,1]$。本文充分考虑密钥扩展算法产生的轮密钥间的制约关系，提出新的 Kiasu-BC 算法的 5 轮中间相遇区分器，得到定理 1。

定理 1 给定 Kiasu-BC 算法的一个 5-δ-集 $\{w_0^0,w_0^1,\cdots,w_0^{31}\}$，将这个集合进行 5 轮 Kiasu-BC 算法加密，满足 $w_0[0]$ 是活跃字节，若该集合中存在一对元素 (w_0^i,w_0^j) 满足图 5 所示的截断差分特征，则相应的有序序列 $(x_6^1[0,1]\oplus x_6^0[0,1],x_6^2[0,1]\oplus x_6^0[0,1],\cdots,x_6^{31}[0,1]\oplus x_6^0[0,1])$ 只有 2^{61} 种可能的取值。

证明 第一步，需要证明有序序列 $(x_6^1[0,1]\oplus x_6^0[0,1],x_6^2[0,1]\oplus x_6^0[0,1],\cdots,x_6^{31}[0,1]\oplus x_6^0[0,1])$ 由一个 5 bit 参数和 25 个字节决定，即

$$w_0^i[0]\|x_2^i[0]\|x_3^i[0,1,2,3]\|k_4^0\|k_5^0[0,5,10,15]$$

记 $x^m\oplus x^i$ $(m=0,1,\cdots,32)$ 为 Δx^m（其中 m 与 i 表示 x 取值对应为不同的元素）。令 $\Delta k_1^1[0]=\Delta w_0^m[0]=w_0^m[0]\oplus w_0^i[0]$，使 $\Delta x_1^m[0]=0$。利用 $\Delta x_2^m[0]=\Delta k_2^1[0]=\Delta k_1^1[0]$ 和已知的 $x_2^i[0]$，可计算出 $\Delta y_2^m[0]=S(x_2^i[0])\oplus S(\Delta x_2^m[0]\oplus x_2^i[0])$。由于算法轮函数中的行移位、列混合、轮调柄密钥加变换都是线性的，故可得 $\Delta x_3^m[0,1,2,3]$。已知 $x_3^i[0,1,2,3]$，那么通过等式 $y_3^m[0,1,2,3]=S(\Delta x_3^m[0,1,2,3]\oplus x_3^i[0,1,2,3])$，可计算出 $y_3^m[0,1,2,3]$。又由于 k_4^0 已知，那么可通过一系列轮函数变换求出 $w_4^m[0,5,10,15]$，然后由已知的 $k_5^0[0,5,10,15]$，可计算出 $x_5^m[0,5,10,15]$。由得到的 $x_5^m[0,5,10,15]$，可求出 $y_5^m[0,5,10,15]$，进而容易得到 $\Delta y_5^m[0,5,10,15]$。同样由于轮函数中的行移位、列混合、轮调柄密钥加变换都是线性的，故可得 $\Delta x_6^m[0,1]$。通过将 $\Delta x_6^m[0,1]$ 和 $\Delta x_6^0[0,1]$ 进行异或运算，可以得到有序序列

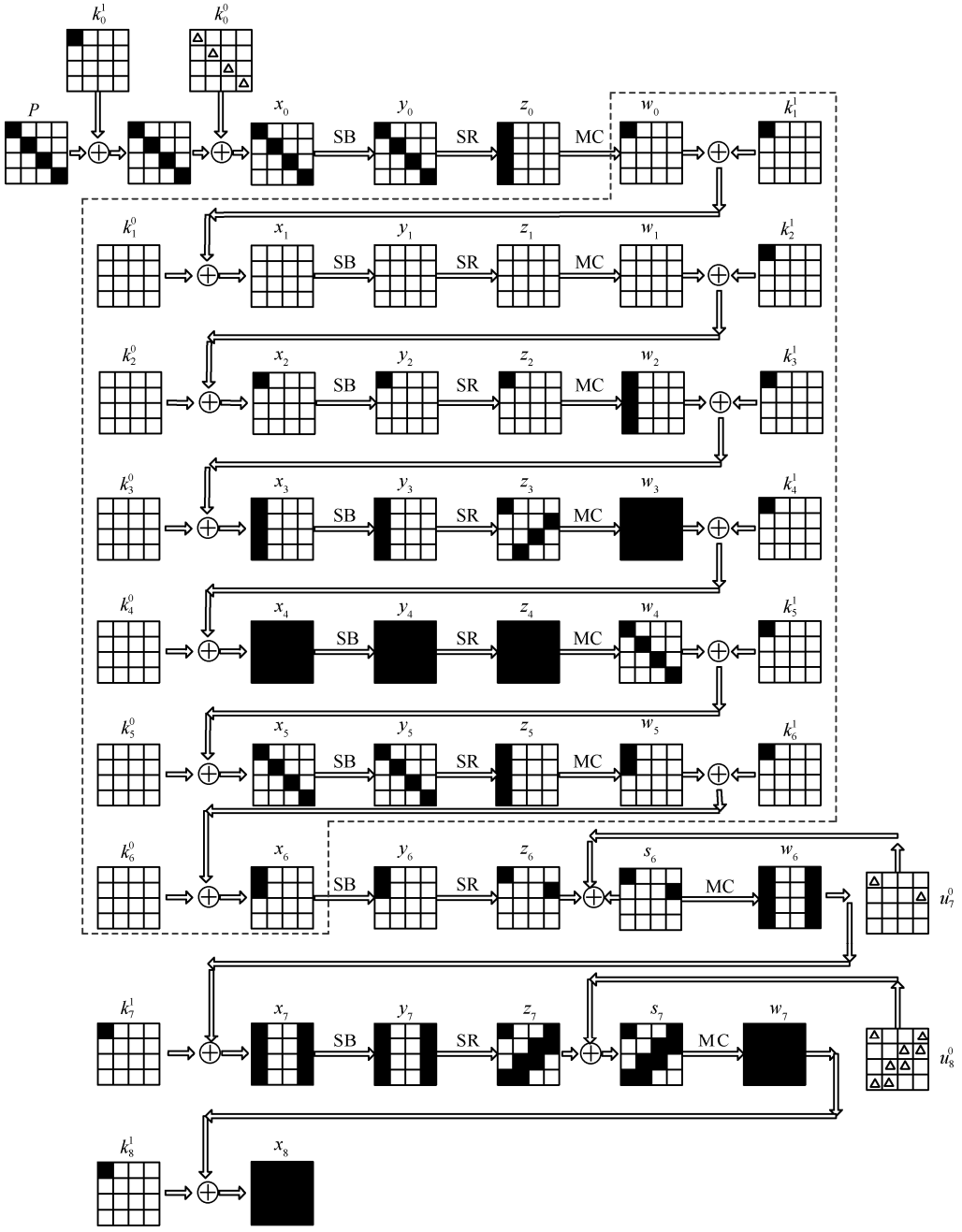

图 5　8 轮 Kiasu-BC 算法中间相遇攻击路径

$[0,1]\oplus x_6^0[0,1], x_6^2[0,1]\oplus x_6^0[0,1],\cdots, x_6^{31}[0,1]\oplus x_6^0[0,1])$。

第二步，需要证明上述参数字节可由一个 5 bit 参数和 11 个字节决定，即

$$w_0^j[0]\|x_2^i[0]\|x_3^i[0,1,2,3]\|y_5^i[0,5,10,15]\|\Delta w_5^j[0,1]$$

由已知的 $w_0^j[0]$ 可以推断出 $\Delta k_1^1[0]$，继而可计算 $\Delta x_2^j[0]=\Delta k_2^1[0]=\Delta k_1^1[0]$。由 $\Delta x_2^j[0]$ 和已知的 $x_2^i[0]$，可以利用等式 $\Delta y_2^j[0]=S(x_2^i[0])\oplus S(\Delta x_2^j[0]\oplus x_2^i[0])$ 计算出 $\Delta y_2^j[0]$，显然，Δy_2^j 的其他 15 个字节差分为 0，故可求出 $\Delta x_3^j[0,1,2,3]$。由 $\Delta x_3^j[0,1,2,3]$ 和已知的 $x_3^i[0,1,2,3]$，可以计算出 $\Delta y_3^j[0,1,2,3]=S(x_3^i[0,1,2,3])\oplus S(\Delta x_3^j[0,1,2,3]\oplus x_3^i[0,1,2,3])$，进而可求出 Δx_4^j。从另一个方向来看，已知 $\Delta w_5^j[0,1]$，且 Δw_5^j 的其他字节都为 0，故可求出 $\Delta y_5^j[0,5,10,15]$。类似地，由于 $y_5^i[0,5,10,15]$ 和 $\Delta y_5^j[0,5,10,15]$ 已知，可以求得 $\Delta x_5^j[0,5,10,15]$，易知 $\Delta w_4^j[0,5,10,15]=\Delta x_5^j[0,5,10,15]$。由于 Δw_4^j 的其他字节都为 0，即可

求得 Δy_4^j。根据性质 1，由 $\Delta x_4^j \| \Delta y_4^j$ 已知，平均可以求出 $x_4^i \| y_4^i$ 的一个解。然后，可以用 $x_3^i[0,1,2,3]$ 和 x_4^i 计算密钥 k_4^0，用 y_4^i 和 $y_5^i[0,5,10,15]$ 计算密钥 $k_5^0[0,5,10,15]$。

这样，就用一个 5 bit 参数和 11 个字节推出了第一步中的字节参数。

此外，结合密钥扩展算法产生的密钥间关系，可以由 k_4^0 计算出 $k_5^0[0,5,10,15]$，不难看出，$k_5^0[0,5,10,15]$ 可以通过 2 种独立的方法来计算[17,25]：一种由截断差分特征直接推断出来，另一种由已知的密钥计算出来。如果 2 种方法计算的结果不相等，那么对应的 93 bit 差分一定是错误的。故有序序列 $(x_6^1[0,1] \oplus x_6^0[0,1], x_6^2[0,1] \oplus x_6^0[0,1], \cdots, x_6^{31}[0,1] \oplus x_6^0[0,1])$ 只有 $2^{5+11\times8-4\times8} = 2^{61}$ 种可能的取值。证毕。

为了确保分析工作的严谨性，一般在选择 δ-集时，如果对于 4×4 状态矩阵的每一块是 8 bit 的算法，可选择 5-δ-集；如果对于 4×4 状态矩阵的每一块是 4 bit 的算法，可选择 4-δ-集。选择不同的 δ-集对数据复杂度的影响比较小。本文的主要工作是发掘密钥间的制约关系来降低存储复杂度，给出截断差分特征的定理描述并进行详细证明。在此基础上，构造 Kiasu-BC 算法新的 5 轮中间相遇区分器，继而实现 Kiasu-BC 算法的 8 轮中间相遇攻击。

4 改进的 8 轮 Kiasu-BC 算法的中间相遇攻击

在一些情况下，为了降低攻击的复杂度，需要交换列混合变换与轮调柄密钥加变换的顺序。因为这 2 个变换都是线性的，可以直接交换它们的顺序，同时参与异或运算的轮密钥相应地变为 $u_i^0 = \mathrm{MC}^{-1}(k_i^0)$。研究密钥扩展算法产生的轮密钥间的制约关系，利用内部密钥间的关系减少区分器中涉及的有序序列的取值个数，构造 5 轮中间相遇区分器，在区分器的前面加一轮、后面加 2 轮，实现对 8 轮 Kiasu-BC 算法进行中间相遇攻击，具体如图 5 所示。改进后的 8 轮 Kiasu-BC 算法中间相遇攻击的时间复杂度为 2^{114}，存储复杂度为 2^{63}，数据复杂度为 2^{108}。

4.1 Kiasu-BC 算法的 8 轮攻击过程

攻击过程可以分为预计算阶段（线下部分）和在线阶段（线上部分）2 个阶段。

预计算阶段。该阶段需计算有序序列 $(x_6^1[0,1] \oplus x_6^0[0,1], x_6^2[0,1] \oplus x_6^0[0,1], \cdots, x_6^{31}[0,1] \oplus x_6^0[0,1])$ 的全部 2^{61} 种可能的取值，并将其存储于哈希表 H 中。

在线阶段。首先需要找到一对满足图 5 所示的截断差分特征的明文。然后构造包含该明文对的 5-δ-集，计算对应的有序序列 $(x_6^1[0,1] \oplus x_6^0[0,1], x_6^2[0,1] \oplus x_6^0[0,1], \cdots, x_6^{31}[0,1] \oplus x_6^0[0,1])$。最后检测计算的序列是否存在于哈希表 H 中。攻击的过程描述如下。

1) 首先定义一个明文集在(0, 5, 10, 15)处遍历，在其他 12 个字节处固定，该集合由 2^{32} 个明文组成。其次定义一个调柄集合在 $T[0]$ 的前 5 bit 遍历，其余字节处固定，集合由 2^5 个调柄组成。利用这 2 个集合可以得到 $2^{4\times8+5} = 2^{37}$ 个密钥调柄组合，进而可得到 $2^{37} \times (2^{37} - 1) \div 2 \approx 2^{73}$ 个满足图 5 所示的截断差分特征的明文差分对。

2) 截断差分特征的概率为 $2^{-(3+1+8+6)\times8} = 2^{-144}$，需要选取 $2^{144-73} = 2^{71}$ 个明文结构，因此共有明文−调柄组合 $2^{71+37} = 2^{108}$ 个。

3) 查询结构中每个明文−调柄组合所对应的密文，并对其进行部分解密，得到 s_7 的值，且满足每一个消息对在 $s_7[1,2,3,4,8,11,14,15]$ 处的差分为 0。满足上面条件的消息对有 $2^{144-8\times8} = 2^{80}$ 对，对 2^{80} 个消息对中的每一个执行以下步骤。

① 由已知的 $w_0[0]$ 可以推断出 $\Delta w_0[0]$ 和 $\Delta k_1^1[0]$，容易得到 $\Delta y_0[0,5,10,15]$。由明文差分计算 $\Delta x_0[0,5,10,15]$，根据性质 1 可得 $x_0[0,5,10,15]$，继而可计算 $k_0^0[0,5,10,15]$。

② 猜测 $\Delta y_6[0,1]$，推断出 $\Delta x_7[0,1,2,3,12,13,14,15]$。由密文计算 $\Delta y_7[0,1,2,3,12,13,14,15]$，故根据性质 1 可得 $x_7[0,1,2,3,12,13,14,15] \| y_7[0,1,2,3,12,13,14,15]$。由 $y_7[0,1,2,3,12,13,14,15]$ 和 $s_7[0,1,2,3,12,13,14,15]$ 可得密钥 $u_8^0[0,1,2,3,12,13,14,15]$。

③ 从消息对中任选一个消息进行部分加密后，构造 5-δ-集，通过部分解密得到 32 个明文−调柄组合 $\{(P^0,T^0),(P^1,T^1),\cdots,(P^{31},T^{31})\}$。

④ 猜测 $u_7^0[0,13]$，查询 32 个明文−调柄组合 $\{(P^0,T^0),(P^1,T^1),\cdots,(P^{31},T^{31})\}$ 得到对应的密文。利用 $u_7^0[0,13]$ 和 $u_8^0[0,1,2,3,12,13,14,15]$ 部分解密密文，得到 $\Delta x_6[0,1]$，进而计算出有序序列 $(x_6^1[0,1] \oplus x_6^0[0,1], x_6^2[0,1] \oplus x_6^0[0,1], \cdots, x_6^{31}[0,1] \oplus x_6^0[0,1])$。

⑤ 检测序列是否存在于哈希表 H 中，若不存在，舍去相应的轮密钥。一个错误密钥通过检测的概率为

$2^{61-31\times16}=2^{-435}$，猜测的密钥为 $2^{80+4\times8}=2^{112}$ 个。所以最后大约有 $1+2^{112-435}\approx1$ 个 $k_0^0[0,5,10,15]$、$u_7^0[0,13]$ 和 $u_8^0[0,1,2,3,12,13,14,15]$ 保留。

4) 穷举 u_8^0 的其他 8 个字节，恢复主密钥。

4.2　复杂度分析

攻击复杂度的分析过程如下。

预计算阶段。每个有序序列有 $31\times16=496\approx2^9$ bit，大约需要 $2^{61}\times2^9\div128=2^{63}$ 个 128 bit 块的存储空间。对 5-δ-集 $\{w_0^0,w_0^1,\cdots,w_0^{31}\}$ 进行部分加密，约等价于 1.5 轮 Kiasu-BC 加密运算，因此预计算阶段的时间复杂度为 $2^{61}\times2^5\times1.5\div9\approx2^{63.6}$ 次 8 轮 Kiasu-BC 加密。

在线阶段。时间复杂度主要由 4.1 节步骤 3) 决定，对 32 个值的计算过程约等价于一轮 Kiasu-BC 加密运算，时间复杂度为 $2^{80}\times2^{32}\times2^5\times1\div8=2^{114}$ 次 8 轮 Kiasu-BC 加密。攻击所需的数据量为 $2^{71+37}=2^{108}$ 个选择明文。

综上可得，该攻击的时间复杂度为预计算阶段时间复杂度与在线阶段时间复杂度之和，约为 2^{114}，存储复杂度为 2^{63}，数据复杂度为 2^{108}。

因此，改进后的中间相遇攻击时间复杂度为 2^{114} 次 8 轮 Kiasu-BC 算法加密，存储复杂度为 2^{63} 个 128 bit 块，数据复杂度为 2^{108} 个明文−调柄组合。

表 2 将本文对 Kiasu-BC 算法的攻击结果与其他中间相遇攻击结果进行了对比。

表 2　Kiasu-BC 算法与其他中间相遇攻击结果的比较

文献	攻击方法	轮数	时间复杂度	存储复杂度	数据复杂度
文献[8]	中间相遇攻击	8	2^{116}	2^{86}	2^{116}
文献[11]	中间相遇攻击	8	$2^{112.8}$	$2^{92.91}$	2^{109}
本文	中间相遇攻击	8	2^{114}	2^{63}	2^{108}

与文献[8]的工作相比，本文工作主要是纠正了文献[8]的中间相遇攻击中所用到的截断差分特征的概率，并基于密钥扩展算法探寻 Kiasu-BC 算法轮密钥间的关系，利用密钥间制约关系改进中间相遇结果，大大降低了存储复杂度；与文献[11]的工作相比，本文充分考虑算法自身的密钥相关性，降低了攻击过程的存储复杂度，同时可以启发有兴趣的研究者深度发掘更多的密钥相关性，利用密钥桥等技术改进现有的攻击结果。深度发掘 Kiasu-BC 算法的密钥间关系，也可以为其他的攻击方法提供更好的改进思路，例如 Kiasu-BC 算法的不可能差分攻击。

5　结束语

本文主要研究了 Kiasu-BC 算法的中间相遇攻击。通过研究 Kiasu-BC 的密钥扩展算法，利用调柄自由度以及内部密钥间的制约关系，结合 Kiasu-BC 算法轮变换中的线性关系，提出了新的 5 轮中间相遇区分器，降低了存储复杂度，对 8 轮 Kiasu-BC 算法中间相遇攻击的现有结果进行了优化。改进后的 8 轮 Kiasu-BC 算法中间相遇攻击的时间复杂度为 2^{114}，存储复杂度为 2^{63}，数据复杂度为 2^{108}。

如何构造较长轮数的中间相遇区分器，是笔者进一步研究的方向；将中间相遇攻击与其他攻击方法相结合，实现更多轮数的攻击路径，从而得到更有效的分析结果，也是笔者将要研究的工作。

参考文献：

[1] LISKOV M, RIVEST R, WAGNER D. Tweakable block ciphers[C]//Advances in Cryptology – CRYPTO 2002. Berlin: Springer, 2002: 31-46.

[2] JEAN J, NIKOLIC I, PEYRIN T. Tweaks and keys for block ciphers: the Tweakey framework[C]//Advances in Cryptology – ASIACRYPT 2014. Berlin: Springer, 2014: 274-288.

[3] JEAN J, NIKOLIĆ I, PEYRIN T. KIASU-submission to the CAESAR competition[EB]. [2018-11-28](2022-02-09).

[4] JEAN J, NIKOLIĆ I, PEYRIN T. Joltik-submission to the CAESAR competition[EB]. [2018-11-28](2022-02-09).

[5] JEAN J, NIKOLIĆ I, PEYRIN T. Submission to CAESAR[EB]. [2016-10](2022-02-09).

[6] DOBRAUNIG C, EICHLSEDER M, MENDEL F. Square attack on 7-round Kiasu-BC[C]//International Conference on Applied Cryptography and Network Security. Berlin: Springer, 2016: 500-517.

[7] ABDELKHALEK A, TOLBA M, YOUSSEF A M. Cryptanalysis of some block cipher constructions[D]. Montreal: The Concordia Institute, 2017.

[8] TOLBA M, ABDELKHALEK A, YOUSSEF A M. A meet in the middle attack on reduced round Kiasu-BC[J]. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 2016, 99(10): 1888-1890.

[9] DOBRAUNIG C, LIST E. Impossible-differential and boomerang cryptanalysis of round-reduced Kiasu-BC[C]//Topics in Cryptology – CT-RSA 2017. Berlin: Springer, 2017: 207-222.

[10] JIANG Z L, JIN C H. Multiple impossible differentials cryptanalysis on 7-round ARIA-192[J]. Security and Communication Networks, 2018, 2018: 7453572.

[11] LIU Y, SHI Y F, GU D W, et al. Improved meet-in-the-middle attacks on reduced-round Kiasu-BC and Joltik-BC[J]. The Computer Journal, 2019, 62(12): 1761-1776.

[12] DIFFIE W, HELLMAN M E. Special feature exhaustive cryptanalysis of the NBS data encryption standard[J]. Computer, 1977, 10(6): 74-84.

[13] DEMIRCI H, SELÇUK A A. A meet-in-the-middle attack on 8-round AES[C]//International Workshop on Fast Software Encryption. Berlin: Springer, 2008: 116-126

[14] GILBERT H, MINIER M. A collisions attack on the 7-rounds Rijndael[C]//AES Candidate Conference. Berlin: Springer, 2000: 1-11.

[15] DUNKELMAN O, KELLER N, SHAMIR A. Improved single-key attacks on 8-round AES-192 and AES-256[C]//Advances in Cryptology - ASIACRYPT 2010. Berlin: Springer, 2010: 158-176.

[16] DERBEZ P, FOUQUE P A, JEAN J. Improved key recovery attacks on reduced-round AES in the single-key setting[C]//Advances in Cryptology – EUROCRYPT 2013. Berlin: Springer, 2013: 371-387.

[17] LI L B, JIA K T, WANG X Y. Improved single-key attacks on 9-round AES-192/256[C]//International Workshop on Fast Software Encryption. Berlin: Springer, 2015: 127-146.

[18] DONG X Y, LI L B, JIA K T, et al. Improved attacks on reduced-round camellia-128/192/256[C]//Lecture Notes in Computer Science. Berlin: Springer, 2015: 59-83.

[19] LIN L, WU W L, ZHENG Y F. Improved meet-in-the-middle distinguisher on Feistel schemes[C]//International Conference on Selected Areas in Cryptography. Berlin: Springer, 2015: 122-142.

[20] BIRYUKOV A, DERBEZ P, PERRIN L. Differential analysis and meet-in-the-middle attack against round-reduced TWINE[C]//International Workshop on Fast Software Encryption. Berlin: Springer, 2016: 3-27.

[21] DERBEZ P, PERRIN L. Meet-in-the-middle attacks and structural analysis of round-reduced PRINCE[C]//International Workshop on Fast Software Encryption. Berlin: Springer, 2015: 190-216.

[22] DERBEZ P, FOUQUE P A. Automatic search of meet-in-the-middle and impossible differential attacks[C]//Advances in Cryptology – CRYPTO 2016. Berlin: Springer, 2016: 157-184.

[23] BIHAM E, SHAMIR A. Differential cryptanalysis of DES-like cryptosystems[J]. Journal of Cryptology, 1991, 4(1): 3-72.

[24] KANDA M. Practical security evaluation against differential and linear cryptanalyses for Feistel ciphers with SPN round function[C]//Selected Areas in Cryptography. Berlin: Springer, 2001: 168-179.

[25] LI R J, JIN C H. Meet-in-the-middle attacks on 10-round AES-256[J]. Designs, Codes and Cryptography, 2016, 80(3): 459-471.

[作者简介]

李曼曼（1986− ），女，河南开封人，博士，信息工程大学讲师，主要研究方向为网络空间安全、信息安全、对称密码的设计与分析等。

陈少真（1967− ），女，江苏无锡人，博士，信息工程大学教授，主要研究方向为密码学与信息安全。