DWB-AES：基于 AES 的动态白盒实现方法

王滨[1,2]，陈思[2]，陈加栋[2]，王星[1,2]
（1. 浙江大学电气工程学院，浙江 杭州 310058；2. 中国电科集团 52 所海康威视网络与信息安全实验室，浙江 杭州 310053）

摘 要：物联网设备因资源受限，需要兼具安全性、灵活性的轻量级密码模块保障安全，白盒密码能够满足物联网设备的安全需求。在常见的白盒密码实现方法中，往往密钥和查找表是绑定的，因此每次更换密钥都需要重新生成并更换查找表，这在实际应用中不够灵活。为了解决该问题，提出了一种基于 AES 的动态白盒实现方法，即 DWB-AES。该方法通过改变轮与轮之间的边界，将加解密过程的所有操作均通过查找表来实现，并对表和密钥分别进行混淆，使整个加解密过程不会暴露密钥信息，且每次更换密钥时不需要更换查找表，所以 DWB-AES 更加灵活和实用。通过对 DWB-AES 的安全性分析表明，DWB-AES 具有较高的白盒多样性和白盒含混度，且能够有效地抵御 BGE 和 Mulder 等常见的白盒攻击方法。

关键词：AES；白盒密码；动态白盒；BGE；查找表
中图分类号：TP393
文献标识码：A
DOI: 10.11959/j.issn.1000−436x.2021020

DWB-AES: an implementation of dynamic white-box based on AES

WANG Bin[1,2], CHEN Si[2], CHEN Jiadong[2], WANG Xing[1,2]
1. College of Electrical Engineering, Zhejiang University, Hangzhou 310058, China
2. Network and Information Security Laboratory of Hikvision, The 52th Research Institute of CETC, Hangzhou 310053, China

Abstract: The resources of IoT devices are limited. Therefore, security, flexibility and lightweight cryptographic modules are required. The idea of white-box cryptography can meet the needs of IoT devices. In common AES white-box implementations, keys are bound to look up tables. So the look up tables must be changed when the keys are changed. It is not flexible enough in practical applications. To solve this problem, a dynamic white-box implementation method for AES, which was called DWB-AES, was proposed. By changing the boundary between rounds, all operations of the encryption and decryption process were performed by looking up the tables, and the tables and the keys were respectively confused, so that the entire encryption and decryption process did not expose the key information. The look up tables need not to be changed every time when the keys changed, which was more practical. The security analysis of DWB-AES shows that the DWB-AES has higher white-box diversity and ambiguous, it can resist BGE attack and Mulder attack.

Keywords: AES, white-box cryptography, dynamic white-box, BGE, look up table

1 引言

由于物联网感知层的设备受到软硬件资源的各种限制，而传统的密码技术需要消耗较多的资源，从而导致传统的密码技术手段很难直接应用到感知层设备。白盒密码作为一种软密码模块，与传统的密码技术不同，能够隐藏密钥信息、极大地提高物联网设备的安全性，同时能够大幅降低成本，非常适用于对成本敏感的感知层设备。

白盒密码颠覆了传统密码学对攻击者能力的

收稿日期：2020−09−15；**修回日期**：2020−12−08
基金项目：国家重点研发计划基金资助项目（No.2018YFB2100400）；国家电网公司总部科技基金资助项目（No.5700-202019187A-0-0-00）
Foundation Items: The National Key Research and Development Program of China (No.2018YFB2100400), Science and Technology Project of State Grid Corporation of China (No.5700-202019187A-0-0-00)

诸多限制，且应用的成本较低，所以更适合应对实际中的安全威胁。目前，白盒密码已经在数字版权保护管理、云上数据软件加解密、移动智能终端信息加密保护等方面得到了较广泛的应用，引起了学术界和产业界的广泛关注。

白盒密码学的概念由 Chow 等[1]提出，同时 Chow 等[2]通过查找表的方式实现了数据加密标准（DES, data encryption standard）和 AES[1]的白盒密码算法。然而，文献[3-4]中提出了对 Chow 等白盒 DES 算法的有效攻击方法。Billet 等[5]给出了对 Chow 等白盒算法的攻击方法，被称为 BGE 攻击。在 BGE 攻击被提出之后，Xiao 等[6]利用更大的线性编码设计了能抵御 BGE 攻击的 AES 白盒密码算法。然而 Mulder 等[7]以 2^{30} 的时间复杂度提取了 Xiao-Lai 白盒算法[6]的密钥。为了抵御 Mulder 等的攻击，文献[8]在 Xiao-Lai 白盒算法的基础上，通过增加非线性编码的方式设计了新的 AES 白盒算法。此外，Karroumi[9]通过引入二元密文的方式实现了 AES 白盒密码算法，但是随后由 Mulder[10]攻破。Biryukov 等[11]提出了基于 ASASA（affine-substitution-affine-substitution-affine）结构的多变量密码体系，并设计了基于 ASASA 结构的白盒密码方案，通过插入扰乱项来抵抗攻击。文献[12]以分组密码算法为底层模块，设计了基于底层分组密码算法安全性[13]的白盒密码。文献[14]设计了类 AES 的白盒算法。文献[15]设计了一种基于 SM4 的白盒算法。此外，近年来还有其他密码算法的白盒实现被研究和设计[16-18]。但是相比于其他白盒实现思路，基于查找表的白盒方案计算相对简单、占用空间较小、计算效率更高。

由于白盒密码是软件实现的，因此可以很好地满足对成本和安全都有较高要求的场景，但是当前白盒密码技术存在以下的问题。

1) 灵活性较差。为了保证加解密的安全性，需要定期更换密钥，而现有的白盒算法的表是与密钥绑定的，即静态白盒算法，每次更换密钥需要重新生成和更换表，甚至更换白盒算法库。

2) 实用性不好。如前所述，已有的白盒算法要么存在安全设计缺陷，容易被敌手攻击；要么为了提高安全性，增加计算复杂度，需要占用较大的存储和计算资源，这对当前白盒密码的应用场景来说不适用。

基于当前的研究现状和需求，本文提出一种基于查找表的动态 AES 白盒算法——DWB-AES，该算法具有以下特点。

1) 具有较高的灵活性。每次更换密钥时，不必更新表和白盒算法库，只需要更新很小的白盒密钥信息即可。

2) 具有较高的实用性。DWB-AES 加解密过程基于查找表实现，具有较高的运算效率和较低的存储空间需求，尤其适合计算和存储资源受限的物联网设备。

3) 具有较高的安全性。DWB-AES 对密钥和表同时混淆，引入 16 B 的混淆变换，使其能够抵御现有针对白盒密码的 BGE 和 Mulder 等常用攻击方法，并且拥有较高的白盒多样性和白盒含混度。

2 AES 静态白盒算法

在白盒攻击环境下，攻击者可以观察到密码算法执行的整个过程，为了保护密钥，可以将加解密过程以表的形式实现，这种思路是由 Chow 等提出的。通过查找表，由输入数据直接获取输出数据，从而隐藏了密钥信息。下面，将分别介绍 Chow 等白盒实现和 Xiao-Lai 白盒算法。

2.1 Chow 等白盒实现

在 Chow 等白盒实现里[1]，通过改变轮与轮之间的边界，将加密钥和 S 盒变换组合在一起，对应一个 8 bit 双射，称为 T 盒变换。以 AES-128 为例，第 r 轮、第 i 行、第 j 列的 T 盒为

$$T_{i,j}^{r}(x)=S\left(x\oplus K_{i,j}^{r}\right)\left(0\leqslant i,j\leqslant 3,1\leqslant r\leqslant 9\right) \quad (1)$$

其中，$K_{i,j}^{r}$ 表示第 r 轮、第 i 行、第 j 列的轮密钥，S 表示 S 盒变换。第 10 轮的 T 盒需要包含最后一轮加密钥操作，即

$$T_{i,j}^{10}(x)=S\left(x\oplus K_{i,j}^{10}\right)\oplus K_{\mathrm{sr}(i,j)}^{11}\left(0\leqslant i,j\leqslant 3\right) \quad (2)$$

其中，$\mathrm{sr}(i,j)$ 表示行变换之后的下标。

列混淆操作每次作用在状态矩阵的一列，对应 32×32 的列混淆矩阵和 32×1 的列向量相乘。为了减小表大小，将列混淆矩阵 $\mathbf{MC}$ 分割成 4 个部分，即 $\mathbf{MC}=(\mathbf{MC}_0,\mathbf{MC}_1,\mathbf{MC}_2,\mathbf{MC}_3)$，将列混淆操作 $\mathbf{MC}$ 转换为 4 次 32×8 矩阵和 8×1 列向量相乘，并对 4 个列向量相加。

为了隐藏密钥，需要引入混淆编码。为此，在 T 盒变换之前引入双射变换，在列混淆操作之后乘

32×32 矩阵 **MB**，并使用和分割矩阵 **MC** 一样的方法分割矩阵 **MB**。

将 T 盒变换和列混淆操作结合，生成 TypeII 表；$\text{TypeII}=\text{MB}\circ\text{MC}\circ T\circ\text{mb}$，其中，mb 表示引入的双射变换，MB 表示引入的线性变换：乘矩阵 **MB**。为了抵消引入的线性变换和 mb 双射变换，引入 TypeIII 表，对应实现 mb 的逆变换和 MB 的逆变换。

此外，为了实现矩阵分割过程中的异或操作，构造异或辅助表 TypeIV，实现了 4 bit 和 4 bit 的异或操作。

2.2　Xiao-Lai 白盒实现

和 Chow 等白盒算法类似，Xiao-Lai 的 AES 白盒实现算法将加密钥操作和 S 盒变换结合在一起生成 T 盒，这里不再赘述。与 Chow 等白盒算法不同，Xiao-Lai 的 AES 白盒实现算法的列混淆操作将 **MC** 分割成 2 个部分，即 $\mathbf{MC}=(\mathbf{MC}_0,\mathbf{MC}_1)$，列混淆操作变为 2 次 32×16 矩阵和 16×1 列向量相乘再相加。

构造表 TMC 为

$$\text{TMC}_i=R_{\frac{i}{2}}\circ\text{MC}_{i\bmod2}\circ\left(T_{2i+1}\,\|\,T_{2i}\right)\circ L_i\left(0\leqslant i\leqslant7\right)\quad(3)$$

其中，T_{2i+1} 和 T_{2i} 表示相邻的 2 个 T 盒变换，R 表示 32×32 矩阵线性变换，L 表示 16×16 矩阵线性变换。将查表得到的 2 个列向量直接相加，得到列混淆的结果。

行变换操作可看作乘 128×128 矩阵的操作，通过乘随机矩阵来实现混淆，最终得到矩阵 $\boldsymbol{M}=\boldsymbol{L}\mathbf{SR}\boldsymbol{R}^{-1}$，其中，$\boldsymbol{L}$ 为 16×16 的矩阵，**SR** 为行变换矩阵，$\boldsymbol{R}$ 为 32×32 的矩阵，$\boldsymbol{M}$ 矩阵既实现了行变换操作，又抵消了 TMC 表中引入的线性变换。为了处理外部编码，第一轮的 $\boldsymbol{M}$ 矩阵引入了输入变换，最后一轮的 $\boldsymbol{M}$ 矩阵引入了输出变换。

3　AES 动态白盒算法 DWB-AES 实现方案

所谓动态白盒算法，就是在密钥更换时，不需要更换查找表，可以动态地更换密钥。其核心思想是，对密钥引入混淆变换得到白盒密钥，每次加密时将白盒密钥和明文一起作为输入参数，进行查找表操作，最终得到输出结果。每次更换密钥后，只需更换白盒密钥。

动态白盒使用流程分为初始化和加（解）密 2 个过程，如图 1 所示。

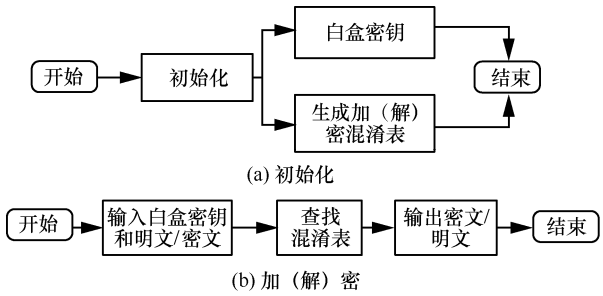


图 1　动态白盒使用流程

初始化过程涉及表的生成和白盒密钥的生成，加（解）密通过查找混淆表实现。

动态白盒算法的实现流程为，对轮密钥和数据进行行变换操作，然后乘 16×16 的非退化矩阵（称该操作为 L 变换），从而实现线性混淆。对混淆后的结果进行异或操作、S 盒变换、列混淆操作，最后乘 32×32 的非退化矩阵（称该操作为 R 操作）。

涉及的表为行变换表（对应行变换和 L 变换）、加密钥表（对应加密钥操作 ARK）、列混淆表（对应 S 盒变换、列混淆操作 MC 和 R 变换）。引入的 L 变换和 R 变换在相邻的表之间抵消。

以 AES-128 为例进行说明，通过改变轮与轮之间的边界，并且将行变换操作提前。修改后的轮边界如图 2 所示。下面，将描述密钥变换和各表的构造过程。

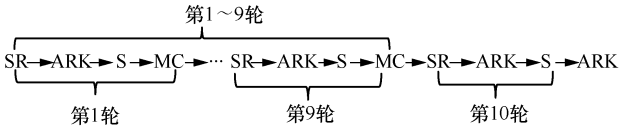


图 2　修改后的轮边界

1) 密钥变换

首先对 AES 密钥进行密钥扩展操作，得到轮密钥。密钥变换的过程可描述如下，以第 r（$1\leqslant r\leqslant10$）轮为例。首先将轮密钥进行变换操作。然后将密钥矩阵的每一列分成两部分，每 2 B 为一组，得到 $\text{GF}(2)^{16}$ 上的列向量，对其进行线性变换：随机生成 16×16 的非退化矩阵 $\boldsymbol{L}_i(i\leqslant0\leqslant7)$，计算

$$\mathbf{LK}_i^r=\boldsymbol{L}_i^r\left(\left(k_{2m,n}^r\right)^{\mathrm{T}},\left(k_{2m+1,n}^r\right)^{\mathrm{T}}\right)^{\mathrm{T}}\left(1\leqslant n\leqslant4,m=0,1\right)\quad(4)$$

其中，乘法操作表示 GF(2) 上的乘法。最后对 $\mathbf{LK}_i^r$ 进行编码，即每 8 bit 为一组进行非线性置换编码，得到 16 B 的白盒密钥。整个流程如图 3 所示。

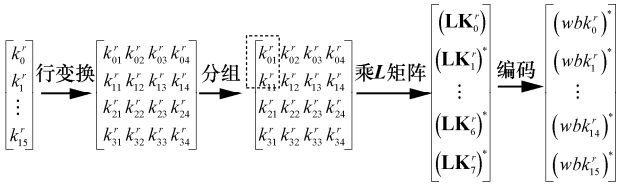


图 3 密钥变换

由于 AES 进行最后一次加密钥操作后不进行行变换，因此，不对最后一次的轮密钥进行行变换操作。对密钥矩阵进行线性变换的过程可以写成

$$\mathbf{LK}=\mathrm{diag}\left(\boldsymbol{L}^0,\boldsymbol{L}^1,\cdots,\boldsymbol{L}^7\right)\left(\left(k_0\right)^{\mathrm{T}},\cdots,\left(k_7\right)^{\mathrm{T}}\right)^{\mathrm{T}} \quad (5)$$

其中，diag 表示对角阵。下面以加密的过程为例，描述查找表的生成方法。按照顺序被查找的表依次为输入变换表、行变换表、异或辅助表、加密钥表、列混淆表和输出变换表。为了更清晰地描述动态白盒实现的过程，将行变换表放到列混淆表后面进行说明，输入输出变换表放到最后进行说明。

2) 加密钥表

加密钥表对应加密钥操作。对于动态白盒实现，密钥是可变的，由于密钥和明文一样是作为参数输入的，若将其嵌入其他表里，则会导致表过于庞大，为此，单独构造 16 bit 输入、8 bit 输出的加密钥表。其中，16 bit 的输入包含 8 bit 白盒密钥和 8 bit 明文。这里的白盒密钥是由密钥变换过程生成的。

表构造过程为，对 8 bit 白盒密钥和 8 bit 明文先解码，再进行异或操作得到 8 bit 的数据；然后对 8 bit 数据进行输出编码：先进行线性变换，再进行非线性编码得到输出数据。加密钥表如图 4 所示。

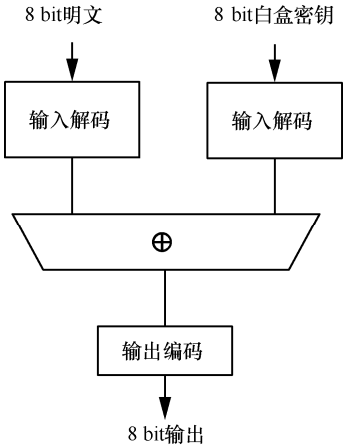


图 4 加密钥表

3) 列混淆表 sTMC

与文献[6]中的 nTMC 表构造思路相同，为了减少表的大小，将列混淆操作分成 2 步，先是将列混淆矩阵 **MC** 分割后和状态矩阵的列相乘，最后将分割相乘的结果再相加，即

$$\mathbf{MC}\begin{bmatrix}\boldsymbol{T}_i^r\\ \boldsymbol{T}_{i+1}^r\end{bmatrix}=\mathbf{MC}_0\boldsymbol{T}_i^r+\mathbf{MC}_1\boldsymbol{T}_{i+1}^r\left(i=0,2,4,6\right) \quad (6)$$

其中，将 **MC** 进行矩阵分块，分成 2 个 32×16 的矩阵，即 $\mathbf{MC}=(\mathbf{MC}_0,\mathbf{MC}_1)$，对状态矩阵分块，得到 $\boldsymbol{T}_i^r$。

列混淆表 sTMC 实现了乘分割后的 **MC** 矩阵：每个 sTMC 表乘 **MC** 的一部分，每轮一共有 8 个 sTMC 表。由于加密钥表的输出数据含有线性变换，因此将 *S* 盒置换嵌入列混淆表里而不是加密钥表，sTMC 表列混淆表的输入为 16 bit，输出为 32 bit。输入的状态矩阵每 2 B 为一组，分成 8 组，分别查找 8 个 sTMC 表，得到 8 个 32 bit 的向量，后面两两一组进行异或，最后得到 128 bit 数据。

表 sTMC_i^r（其中 r 表示轮数，i 表示编号，sTMC_i^r 对应 T_i^r，$0\leqslant i\leqslant 7$，$1\leqslant r\leqslant 9$）的构造过程为，首先对 16 bit 的输入进行解码（每 8 bit 为一组），去掉非线性编码的部分，再乘 16×16 的矩阵 $(\boldsymbol{L}_i^r)^{-1}$，去掉线性编码的部分，再进行 *S* 盒变换；然后乘 $\mathbf{MC}_{i\bmod 2}$ 得到 32 bit 的数据，对该数据乘 32×32 的非退化矩阵 $\boldsymbol{R}_{i/2}^r$，进行线性编码得到输出；最后对 32 bit 的输出每 4 bit 一组进行非线性输出编码。列混淆表的构造过程如图 5 和图 6 所示，这里需要注意的是，列混淆操作只进行 9 轮，因此第 10 轮不再引入列混淆操作，第 10 轮的输出变换矩阵为 $\boldsymbol{L}^{11}$，对应的逆变换在输出编码表中实现。

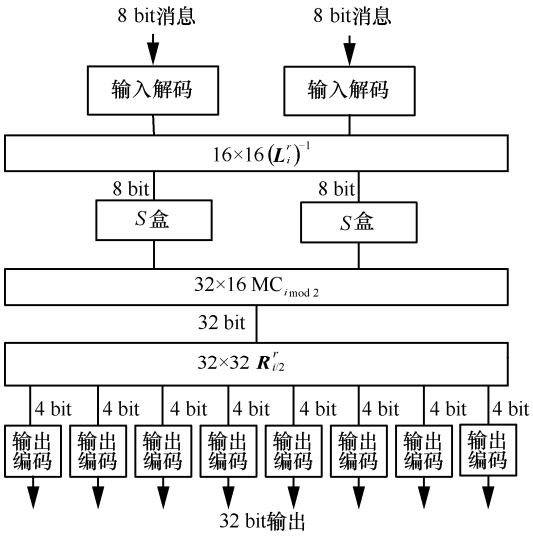


图 5 列混淆表第 1～9 轮

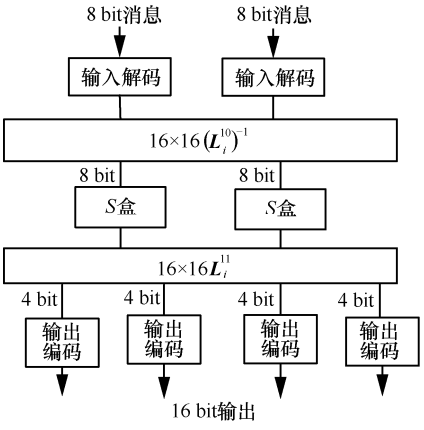


图 6　列混淆表第 10 轮

4) 行变换表

行变换操作可以看作对一个 128 bit 的列向量乘128×128 矩阵的变换，行变换操作可使用查找表的方式来实现。其涉及的线性变换如式(7)所示。

$$\boldsymbol{s}=\begin{bmatrix}\boldsymbol{L}_0^r & & & \\ & \boldsymbol{L}_1^r & & \\ & & \ddots & \\ & & & \boldsymbol{L}_7^r\end{bmatrix}\mathbf{SR}\cdot \begin{bmatrix}\left(\boldsymbol{R}_0^{r-1}\right)^{-1} & & & \\ & \left(\boldsymbol{R}_1^{r-1}\right)^{-1} & & \\ & & \left(\boldsymbol{R}_2^{r-1}\right)^{-1} & \\ & & & \left(\boldsymbol{R}_0^{r-1}\right)^{-1}\end{bmatrix}\begin{bmatrix}b_0^r\\ b_1^r\\ \vdots\\ b_{127}^r\end{bmatrix} \tag{7}$$

若不进行矩阵分割，行变换表大小为 $2^{128}\times128$ bit，这在实际中不可行。因此需要进行分割，$\mathrm{GF}(2)^{128}$ 上的任一列向量 $\boldsymbol{X}$ 可分割成 32 块，其中每块为 $\mathrm{GF}(2)^4$ 上的列向量。即 $\boldsymbol{X}^{\mathrm{T}}=(\boldsymbol{x}_0^{\mathrm{T}},\boldsymbol{x}_1^{\mathrm{T}},\cdots,\boldsymbol{x}_{31}^{\mathrm{T}})^{\mathrm{T}}$，$\boldsymbol{x}_j\left(0\leqslant j\leqslant31\right)$ 为 $\mathrm{GF}(2)^4$ 上的列向量，则式(8)成立。

$$\boldsymbol{S}=\boldsymbol{M}\boldsymbol{X}=\sum_{j=0}^{31}\boldsymbol{M}_j\boldsymbol{x}_j \tag{8}$$

其中，$\boldsymbol{M}=\left(\boldsymbol{M}_0,\boldsymbol{M}_1,\cdots,\boldsymbol{M}_{31}\right)$ 为128×128 的矩阵，$\boldsymbol{M}_j(0\leqslant j\leqslant31)$ 为128×4 的矩阵。

根据上述分割思路，可将一个 $2^{128}\times128$ bit 的表分割成32 个 $2^4\times128$ bit 的表。下面进行具体说明。

令 $\boldsymbol{L}^r=\mathrm{diag}(\boldsymbol{L}_0^r,\boldsymbol{L}_1^r,\cdots,\boldsymbol{L}_7^r)$，则有

$$\widetilde{\boldsymbol{R}}^{r-1}=\mathrm{diag}\left(\left(\boldsymbol{R}_0^{r-1}\right)^{-1},\left(\boldsymbol{R}_1^{r-1}\right)^{-1},\left(\boldsymbol{R}_2^{r-1}\right)^{-1},\left(\boldsymbol{R}_3^{r-1}\right)^{-1}\right)=\mathrm{diag}\left(\widetilde{\boldsymbol{R}}_0^{r-1},\widetilde{\boldsymbol{R}}_1^{r-1},\widetilde{\boldsymbol{R}}_2^{r-1},\widetilde{\boldsymbol{R}}_3^{r-1}\right) \tag{9}$$

令 $\boldsymbol{X}$ 为 $\mathrm{GF}(2)^{128}$ 上的列向量。将 $\boldsymbol{X}^{\mathrm{T}}$ 表示为 $\boldsymbol{X}^{\mathrm{T}}=(\boldsymbol{x}_{0,0}^{\mathrm{T}},\cdots,\boldsymbol{x}_{0,7}^{\mathrm{T}},\boldsymbol{x}_{1,0}^{\mathrm{T}},\cdots,\boldsymbol{x}_{1,7}^{\mathrm{T}},\boldsymbol{x}_{2,0}^{\mathrm{T}},\cdots,\boldsymbol{x}_{2,7}^{\mathrm{T}},\boldsymbol{x}_{3,0}^{\mathrm{T}},\cdots,\boldsymbol{x}_{3,7}^{\mathrm{T}})^{\mathrm{T}}$，可以得到

$$\begin{aligned}
&\boldsymbol{S}=\boldsymbol{L}^r\mathbf{SR}\widetilde{\boldsymbol{R}}^{r-1}\boldsymbol{X}=\\
&\boldsymbol{L}^r\mathbf{SR}\left(\left(\widetilde{\boldsymbol{R}}_0^{r-1}\left(\boldsymbol{x}_{0,0}^r,\cdots,\boldsymbol{x}_{0,7}^r\right)^{\mathrm{T}}\right)^{\mathrm{T}},\left(\widetilde{\boldsymbol{R}}_1^{r-1}\left(\boldsymbol{x}_{1,0}^r,\cdots,\boldsymbol{x}_{1,7}^r\right)^{\mathrm{T}}\right)^{\mathrm{T}}\right)=\\
&\left(\left(\widetilde{\boldsymbol{R}}_2^{r-1}\left(\boldsymbol{x}_{0,0}^r,\cdots,\boldsymbol{x}_{2,7}^r\right)^{\mathrm{T}}\right)^{\mathrm{T}},\left(\widetilde{\boldsymbol{R}}_3^{r-1}\left(\boldsymbol{x}_{3,0}^r,\cdots,\boldsymbol{x}_{3,7}^r\right)^{\mathrm{T}}\right)^{\mathrm{T}}\right)^{\mathrm{T}}=\\
&\boldsymbol{L}^r\mathbf{SR}\left(\left(\sum_{m=0}^{7}\widetilde{\boldsymbol{R}}_{0,m}^{r-1}\boldsymbol{x}_{0,m}^r\right)^{\mathrm{T}},\left(\sum_{m=0}^{7}\widetilde{\boldsymbol{R}}_{1,m}^{r-1}\boldsymbol{x}_{1,m}^r\right)^{\mathrm{T}}\right)=\\
&\left(\left(\sum_{m=0}^{7}\widetilde{\boldsymbol{R}}_{2,m}^{r-1}\boldsymbol{x}_{2,m}^r\right)^{\mathrm{T}},\left(\sum_{m=0}^{7}\widetilde{\boldsymbol{R}}_{3,m}^{r-1}\boldsymbol{x}_{3,m}^r\right)^{\mathrm{T}}\right)^{\mathrm{T}}=\\
&\boldsymbol{L}^r\left(\mathbf{SR}_0\sum_{m=0}^{7}\widetilde{\boldsymbol{R}}_{0,m}^{r-1}\boldsymbol{x}_{0,m}^r\oplus\mathbf{SR}_1\sum_{m=0}^{7}\widetilde{\boldsymbol{R}}_{1,m}^{r-1}\boldsymbol{x}_{1,m}^r\oplus\right.\\
&\left.\mathbf{SR}_2\sum_{m=0}^{7}\widetilde{\boldsymbol{R}}_{2,m}^{r-1}\boldsymbol{x}_{2,m}^r\oplus\mathbf{SR}_3\sum_{m=0}^{7}\widetilde{\boldsymbol{R}}_{3,m}^{r-1}\boldsymbol{x}_{3,m}^r\right)=\\
&\sum_{n=0}^{3}\sum_{m=0}^{7}\boldsymbol{L}^r\mathbf{SR}_n\widetilde{\boldsymbol{R}}_{n,m}^{r-1}\boldsymbol{x}_{n,m}^r
\end{aligned}\tag{10}$$

其中，$\widetilde{\boldsymbol{R}}_{i,j}^{r-1}$ 为32×4 的矩阵，是对矩阵 $\widetilde{\boldsymbol{R}}^{r-1}$ 分块后的矩阵，$\widetilde{\boldsymbol{R}}_i^{r-1}=(\widetilde{\boldsymbol{R}}_{i,0}^{r-1},\widetilde{\boldsymbol{R}}_{i,1}^{r-1},\cdots,\widetilde{\boldsymbol{R}}_{i,7}^{r-1})$；$\boldsymbol{x}_{i,j}$ 为 4 bit 列向量；$\mathbf{SR}_i\left(0\leqslant i\leqslant3\right)$ 为 128×32 的矩阵，且 $\mathbf{SR}=(\mathbf{SR}_0,\mathbf{SR}_1,\mathbf{SR}_2,\mathbf{SR}_3)$。经过分割后，128 bit 列向量和128×128 矩阵的乘法可以看作 32 个 4 bit 列向量分别和128×4 矩阵做乘法再相加。

下面是行变换表的构造过程，行变换表的输入为 8 bit，输出为 128 bit。

输入的 8 bit 数据，其中 4 bit 来自乘 $\mathbf{MC}_0$ 的结果，另外 4 bit 来自乘 $\mathbf{MC}_1$ 的结果。以状态矩阵的第一列为例，其中 4 bit 是查找表 sTMC_0^r 之后的结果，另外 4 bit 是查找 sTMC_1^r 之后的结果。在行变换表里，首先对输入数据做非线性变换，进行解码操作，将来源不同的 4 bit 数据异或，得到 4 bit 列向量；然后按照上述计算式，先乘32×4 的矩阵，抵消列混淆表里的线性编码，再乘128×32 的矩阵，进行行变换；最后乘128×128 的矩阵，进行线性变换，得到 128 bit 的列向量，其中每 4 bit 一组，进行非线性编码，得到输出结果。表构造的过程如图 7 和图 8 所示，在查找第 1 轮 TSR 之前，先进行输入线性编码（乘大小为128×128 随机矩阵 $\mathbf{IN}$），因此第一轮表的线性逆变换乘的是矩阵 $\mathbf{IN}^{-1}$。

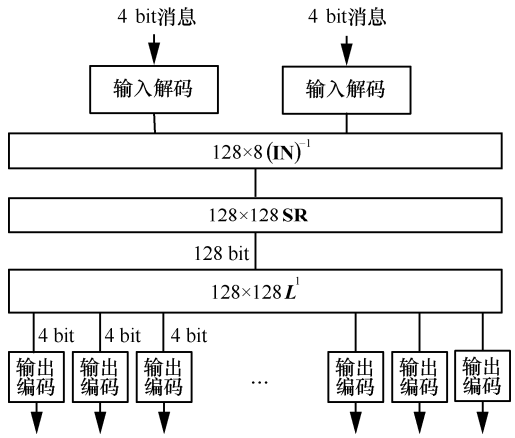


图 7　行变换表第 1 轮

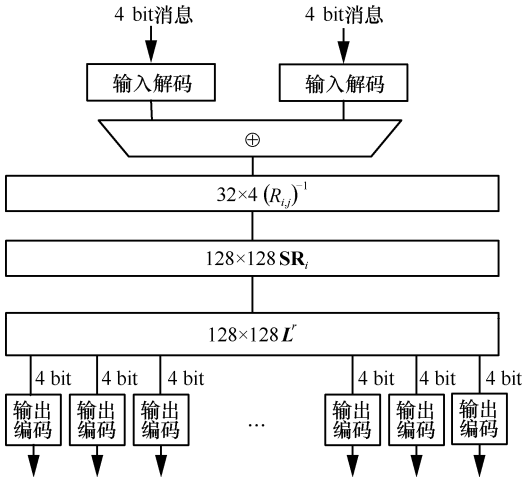


图 8　行变换表第 2～10 轮

5) 异或辅助表 TXOR

行变换的矩阵乘操作被分割成 32 个列向量分别乘分割后的矩阵后再相加。本文用异或辅助表实现异或操作，一共有 32 个乘法操作，对应 32 个行变换表，每个表输出 128 bit 列向量，因此行变换辅助表实现 32 个 128 bit 列向量的加法操作，对应 31 次异或操作。为了减少表大小，将 31 次 128 bit 异或操作分成 31×32 次 4 bit 异或操作。异或辅助表如图 9 所示。

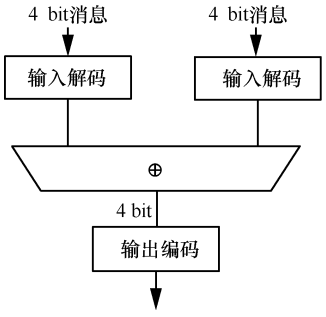


图 9　异或辅助表

6) 输入输出编码表

输入输出编码均是外部编码，输入编码是对输入明文进行乘128×128 非退化矩阵 **IN**，输出编码是对输出前的密文乘128×128 非退化矩阵 **OUT**，这里有

$$\mathbf{OUT}=\left(\boldsymbol{L}^{11}\right)^{-1}=\mathrm{diag}\left(\left(\boldsymbol{L}_0^{11}\right)^{-1},\left(\boldsymbol{L}_1^{11}\right)^{-1},\cdots,\left(\boldsymbol{L}_7^{11}\right)^{-1}\right) \tag{11}$$

因此输入输出编码表实现了128×128 的矩阵乘法，为了减少表大小，对矩阵进行分块，最终使用 16 个规模为8×128 的表来实现。表构造如图 10 所示。

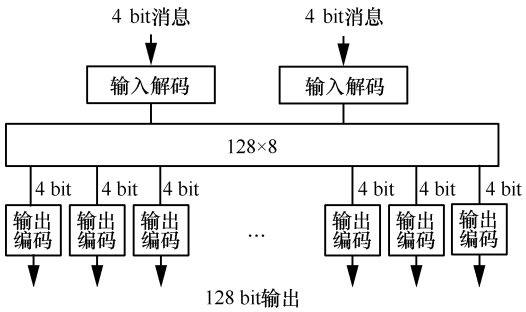


图 10　输入输出编码表

整个加密过程为，首先查找输入编码表和辅助异或表实现输入编码，再重复 10 轮如下的表查找：查找行变换表和辅助异或表实现行变换，查找加密钥表实现加密钥，查找列混淆表实现列混淆和 S 盒变换。10 轮之后，查找加密钥表实现最后一轮加密钥，最后进行输出编码变换得到真正的密文。

4　正确性和效率分析

第 3 节详细介绍了动态白盒实现方案，把 AES 实现的各步骤都使用查找表来实现，每个表的输入和输出都引入了线性变换和非线性变换来实现混淆，为了保证算法的正确性，在相邻表之间引入的变换都是互逆的，即保证了在进行 AES 的各步骤之前的数据都是正确的。为了实现动态白盒算法，预先对轮密钥进行线性和非线性变换，其中非线性变换在加密钥表的输入里被抵消，而线性变换则和输入数据的线性变换相同，并在列混淆表中被抵消。各表的构造示意和描述均在第 3 节给出说明。

下面，将分析动态白盒算法的效率。

TSR 第 1 轮表的大小为 $2^8\times128$ bit，表个数为 16 个，对应 TXOR 表个数为 480 个；第 2～10 轮，每轮需要 32 个表，表大小为 $2^4\times128$ bit，对应 TXOR

表 1　　3 种基于查找表的白盒方案的效率对比

白盒方案	表大小/KB	矩阵乘法次数/次	每轮查表及异或次数/次	更换密钥时需要更换的查找表大小/KB
Chow	752	0	205	752
Xiao-Lai	20 502	11	12	20 502
DWB-AES	32 280	0	1 048	0

表个数为 992 个。sTMC 表第 1～9 轮表大小为 $2^{16}\times32$ bit，每轮需要 8 个表；第 10 轮表大小为 $2^{16}\times16$ bit，需要 8 个表。TK 表大小为 $2^{16}\times8$ bit，每轮需要 16 个表。输入输出编码表大小为 $2^8\times128$ bit，表个数为 32 个，对应 TXOR 表个数为 960 个。各表所占空间总大小如下。

TSR：$16\times2^8\times128+9\times32\times2^4\times128$ bit = 136 KB

TXOR：$(480+9\times992+960)\times2^8\times4$ bit=1 296 KB

sTMC：$9\times8\times2^{16}\times32+8\times2^{16}\times16$ bit=19 546 KB

TK：$11\times2^{16}\times8\times16$ bit=11 264 KB

输入输出编码：$32\times2^8\times128$ bit=128 KB

因此，整个加密过程中所有表的大小为 $136+1\,296+19\,456+11\,264+128=32\,280$ KB。

3 种基于查找表的白盒方案的效率对比如表 1 所示。在 3 种白盒实现里，只有 Xiao-Lai 的白盒实现涉及了耗时多的矩阵乘法，其余均为简单的查找表操作。考虑到空间大小，DWB-AES 所占用的空间是最大的，但仍在可接受范围内，并且由于 DWB-AES 是动态白盒，每次更换密钥需要更换查找表的大小为 0。

此外，3 种基于查找表的白盒实现里，只有 DWB-AES 既能抵御 BGE 攻击又能抵御 Mulder 等的攻击，这将在后面的安全性分析中进行说明。

5　安全性分析

下面，使用常用的分析方法对 DWB-AES 进行安全性分析型。

5.1　本地安全性

查找表技术通过将密钥隐藏在表中，对表进行混淆，最终保证表空开后也无法对外泄露信息[1]，并且保证任何一个表都不会泄露额外信息，也就是所谓的本地安全性。DWB-AES 生成每个表的过程中，都引入了随机的线性变换和非线性编码，每一个查找表都独立引入了随机混淆信息，因此所有的查找表都是本地安全的。

5.2　白盒多样性和白盒含混度

Chow 等提出了评估白盒安全性的 2 个指标，即白盒多样性和白盒含混度。白盒多样性用来评估可选择的混淆表的数量；白盒含混度用来评估给定表可选择的混淆编码的数量。白盒多样性越大，含混度越高，则密钥混淆的越充分，越不容易被提取密钥。

GF(2) 上的 $n\times n$ 阶可非退化矩阵的个数为 $\prod_{i=1}^{n-1}\left(2^n-2^i\right)$，$m\times n$ 列满秩矩阵的个数为 $2^{n(m-n)}\prod_{i=1}^{n-1}\left(2^n-2^i\right)$。例如 GF(2) 上的 4×4 非退化矩阵个数为 20 160，由此计算本文白盒实现中表的白盒多样性。

TK：$\left(2^8!\right)^3\approx2^{5\,052}$

TSR^1：$\left(2^4!\right)^{34}\times20\,160^{64}\times\left(2^{245}\right)^8\approx2^{4\,452}$

TSR^r：$\left(2^4!\right)^{34}\times2^{126}\times\left(2^{245}\right)^8\approx2^{3\,663}$

sTMC^r：$\left(2^8!\right)^2\times\left(2^4!\right)^8\times2^{254}\times2^{1\,023}\approx2^{4\,997}$

sTMC^{10}：$\left(2^8!\right)^2\times\left(2^4!\right)^4\times2^{254}\times2^{1\,023}\approx2^{4\,821}$

TXOR：$\left(2^4!\right)^3\approx2^{132}$

TOUT：$\left(2^4!\right)^{34}\times20\,160^{64}\approx2^{2\,492}$

表 2 给出了 3 种白盒方案的白盒含混度和白盒多样性。TK 的白盒含混度为 $2^8!\times2^8\approx2^{1\,692}$，TXOR 的白盒含混度为 $2^4!\times2^4\approx2^{48}$，其余表的白盒含混度计算起来比较复杂，本文仅给出粗略的下界，TSR 为 $(2^4!)^2\times2^{126}\approx2^{214}$，sTMC 为 $(2^8!)^2\times2^{254}\approx2^{3\,622}$，TOUT 为 $(2^4!)^2\times20\,160^{32}\approx2^{536}$。

表 2　3 种白盒方案的白盒含混度和白盒多样性

白盒方案	白盒含混度	白盒多样性
Chow	2^{114}	2^{769}
Xiao-Lai	2^{271}	$2^{1\,294}$
DWB-AES	$2^{1\,692}$	$2^{5\,052}$

从表 2 可以看出，DWB-AES 拥有较高的白盒含混度和白盒多样性，因此具有较高的安全性。

5.3 抵御 BGE 攻击

Billet 等提出了对 Chow 等的白盒 AES 算法的攻击方法，被称为 BGE 攻击。在 Chow 的白盒设计里，表满足了比较高的白盒多样性和白盒含混度，从单个表里提取密钥信息十分困难，然而由于表与表之间的混淆会相互抵消，将表组合在一起分析能更容易地提取密钥信息。在 BGE 攻击中，将每轮的各变换当作一个整体，看作一个变换，如图 11 所示。从图 11 可以看出，每轮的输入编码和前一轮的输出编码是互逆的，即 $Q_{i,j}^r=(P_{i,j}^{r+1})^{-1}$。

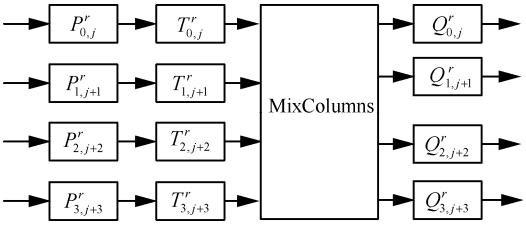


图 11 Chow 等白盒算法流程

BGE 攻击的步骤如下。首先，恢复 $Q_{i,j}^r$ 里的非仿射变换，将 y_0 看作输入 (x_0,x_1,x_2,x_3) 的函数，通过设置 x_1,x_2,x_3 为常数值，可得到函数关系 $y_0(x,c_1,c_2,c_3)\circ y_0(x,c_1,c_2,c_3^*)^{-1}=Q_0(Q_0^{-1}\oplus\beta)$，进而恢复 $Q_{i,j}^r$ 里的非仿射变换。

其次，计算出仿射变换，该步骤主要基于任意的 (y_i,y_j) 存在线性关系，即 $y_i(x_0,00,00,00)=L_{ij}(y_j(x_0,00,00,00))\oplus c_{ij}$。

最后，由 $Q_{i,j}^r$ 得到 $P_{i,j}^{r+1}$，并抽取密钥。

下面，将说明 DWB-AES 可以抵御 BGE 攻击。与 Chow 等白盒算法不同，DWB-AES 的算法实现里，每一轮的置换编码是 16 bit，(y_i,y_j) 不一定存在线性关系。显然，将 TK、sTMC 和 TSR 组合在一起分析，是最容易攻击的。令输入数据为 $(x_0,x_1,\cdots,x_{15})$，令状态矩阵的排列方式为按列排列。将第一轮 TK 表数据输入的第 i 个字节定义为 x_i，将第二轮行变换输出（查找 TSR 和 XOR 表的输出）的第 i 个字节定义为 y_i。令

$$\begin{aligned}&\left(u_{2t}\,\|\,u_{2t+1}\right)^{\mathrm{T}}=\left(\boldsymbol{L}_t^1\right)^{-1}\left(x_{2t}\,\|\,x_{2t+1}\right)^{\mathrm{T}}\\&\left(y_{2t}\,\|\,y_{2t+1}\right)^{\mathrm{T}}=\boldsymbol{L}_t^2\left(v_{2t}\,\|\,v_{2t+1}\right)^{\mathrm{T}}\end{aligned}\tag{12}$$

其中，$t=0$或1，v 是 u 经过加密钥、S 盒变换、列混淆、行变换之后的结果。以前 2 B 为例，第 1 B 的 v_0 和 v_1 的计算式为

$$\begin{aligned}v_0=&\,'02'\otimes S\left(k_0^1\oplus u_0\right)\oplus'03'\otimes S\left(k_1^1\oplus u_1\right)\oplus\\&'01'\otimes S\left(k_2^1\oplus u_2\right)\oplus'01'\otimes S\left(k_3^1\oplus u_3\right)\\v_1=&\,'01'\otimes S\left(k_4^1\oplus u_4\right)\oplus'03'\otimes S\left(k_5^1\oplus u_5\right)\oplus\\&'01'\otimes S\left(k_6^1\oplus u_6\right)\oplus'02'\otimes S\left(k_7^1\oplus u_7\right)\end{aligned}\tag{13}$$

将 16×16 的矩阵 $\boldsymbol{L}_t^2$ 表示为

$$\boldsymbol{L}_t^2=\begin{bmatrix}\left(\boldsymbol{L}_t^2\right)_0&\left(\boldsymbol{L}_t^2\right)_1\\\left(\boldsymbol{L}_t^2\right)_2&\left(\boldsymbol{L}_t^2\right)_3\end{bmatrix}\tag{14}$$

则有

$$\begin{aligned}y_0&=\left[\left(\boldsymbol{L}_0^2\right)_0\left(\boldsymbol{L}_0^2\right)_1\right]\left(v_0\,\|\,v_1\right)^{\mathrm{T}}\\y_1&=\left[\left(\boldsymbol{L}_0^2\right)_2\left(\boldsymbol{L}_0^2\right)_3\right]\left(v_2\,\|\,v_3\right)^{\mathrm{T}}\end{aligned}\tag{15}$$

由于 $\left(\boldsymbol{L}_t^2\right)_s\,(0\leqslant s\leqslant3)$ 不一定可逆，因此 y_0 和 y_1 之间不一定存在线性关系，又由于 $\left(y_{2t}\,\|\,y_{2t+1}\right)^{\mathrm{T}}$ 和 $\left(y_{2t}\,\|\,y_{2t+1}\right)^{\mathrm{T}}$ 自变量不同，故二者之间不一定存在线性关系，即使存在线性关系，未知的加密钥使攻击者无法获得该线性关系与 $\boldsymbol{L}_t^r$ 的关系，因此 BGE 攻击不成立。

5.4 抵御 Mulder 攻击

Mulder 等[7]对 Xiao-Lai 白盒实现进行了攻击，该攻击方法是基于 Biryukow 等[19]的 LE 算法。LE 算法给出了寻找线性变换对 (A,B)，使 $S_2=B\circ S_1\circ A$ 的算法，其中 S_1 和 S_2 均为 n bit 双射，若存在这样的线性变换对，则 S_1 和 S_2 是线性等效的，此时 LE 算法输出符合要求的线性变换对集合；否则，输出相应信息表示 S_1 和 S_2 不是线性等效的。在 Xiao-Lai 白盒实现里，由于 TMC 表只引入了线性变换

$$\begin{aligned}\mathrm{TMC}_i^{(1,j)}=&\,R^{(1,j)}\circ\mathrm{MC}_i\circ(S,S)\circ\oplus\\&\left(k_{2t}^{(1,j)}\,\|\,k_{2t+1}^{(1,j)}\right)\circ\left(\boldsymbol{L}_i^{(1,j)}\right)^{-1}\end{aligned}\tag{16}$$

其中，(S,S) 为 AES 的 S 盒变换。因此若找到 TMC 和 (S,S) 之间的线性变换 (A,B)，则可将表中引入的线性变换去掉，进而提取密钥。

Mulder 等通过观察 Xiao-Lai 白盒实现结构，对 LE 算法进行改造，以便能在 2^{32} 的复杂度内提取密

钥。首先，Mulder 等将密钥相关的 $\mathrm{TMC}_i^{(1,j)}$ 表转换为密钥无关的 $\overline{\mathrm{TMC}}_i^{(1,j)}$ 表。其次，找到线性矩阵 $(\boldsymbol{L}_i^{(1,j)})^{-1}$。最后，提取 AES 密钥。其中，找到线性矩阵的步骤是核心步骤，通过将第 1 轮的 TMC 表和第 2 轮的列混淆操作结合，并将列混淆操作相关输出设置为 0，攻击者可以构造出线性等价解的集合。

下面将说明这种攻击方法不适用于 DWB-AES。由于表的输入和输出均增加了非线性编码，因此使用基于 LE 算法及 Mulder 等的改造算法来寻找线性变换对(A, B)的方法不能直接被应用。即使将非线性编码的部分去掉，由于 TSR 表的输入分别来源于列混淆矩阵 **MC** 和 TMC_{i+1}，线性等价的解集合的构造将更加复杂。

$$\mathrm{mc}_{l,0}\otimes\overline{S}(u_0)\oplus\mathrm{mc}_{l,1}\otimes\overline{S}(u_1)\oplus\mathrm{mc}_{l,2}\otimes\overline{S}(u_2)\oplus\mathrm{mc}_{l,3}\otimes\overline{S}(u_3)=0 \tag{17}$$

令

$$u_0=f_l^t(u_1,u_2,u_3)=\left(\overline{S}\right)^{-1}\left[\mathrm{mc}_{l,0}{}^{-1}\otimes\left(\mathrm{mc}_{l,1}\otimes\overline{S}(u_1)\oplus\mathrm{mc}_{l,2}\otimes\overline{S}(u_2)\oplus\mathrm{mc}_{l,3}\otimes\overline{S}(u_3)\right)\right] \tag{18}$$

其中，$\mathrm{mc}_{l,i}\left(0\leqslant l,i\leqslant 3\right)$ 表示列混淆矩阵的元素，$\overline{S}=S\circ\oplus_{52}$，$S$ 为 AES 的 S 盒，构造出的解集为

$$S_l^t=\left\{x=(x_0\,\|\,x_1)\in\mathrm{GF}\left(2^{32}\right)\mid L_{2t}(x_0)=u_0\,\|\,u_1\wedge L_{2t+1}(x_1)=u_2\,\|\,u_3\wedge u_0=f_l^t(u_1,u_2,u_3)\right\} \tag{19}$$

该集合的大小为 2^{24}，而不是原来的 2^8，这将攻击复杂度提升至 2^{64}，对于使用白盒密码的应用程序来说，可认为是安全的复杂度。

本文使用常用的方法对 DWB-AES 进行了安全行分析，并对 Chow 白盒方案、Xiao-Lai 白盒方案和 DWB-AES 进行了对比，结果如表 3 所示。

表 3　　3 种白盒方案对比

白盒方案	抵御 BGE 攻击	抵御 Mulder 等攻击
Chow	×	—
Xiao-Lai	√	×
DWB-AES	√	√

6　结束语

针对物联网设备软硬件资源有限，密码模块既要满足安全性，又要具有较大灵活性的需求，本文提出了一种不需要更换查找表就能更换密钥的动态白盒实现方法 DWB-AES，在保证正确性的前提下，具有较高的安全性，可以有效地抵御 BGE 攻击和 Mulder 等针对白盒算法的已知常用攻击，由于该方法在密钥更新时不需要更换查找表就能更新密钥，因此应用模式更加灵活，可以更好地满足物联网设备在安全应用中的需求。

参考文献：

[1] CHOW S, EISEN P A, JOHNSON H, et al. White-box cryptography and an AES implementation[C]//Selected Areas in Cryptography. Berlin: Springer, 2003: 250-270.

[2] CHOW S, EISEN P, JOHNSON H, et al. A white-box DES implementation for DRM applications[C]//ACM Workshop on Digital Rights Management. New York: ACM Press, 2003: 1-5.

[3] JACOB M, BONEH D, FELTEN E. Attacking an obfuscated cipher by injecting faults[C]//Digital Rights Management. Berlin: Springer, 2003: 16-31.

[4] GOUBIN L, MASEREEL J M, QUISQUATER M. Cryptanalysis of white box DES implementations[J]. Lecture Notes in Computer Science, 2007, 4876: 278-295.

[5] BILLET O, GILBERT H, ECH-CHATBI C. Cryptanalysis of a white box AES implementation[C]//Selected Areas in Cryptography. Berlin: Springer, 2005: 227-240.

[6] XIAO Y, LAI X. A secure implementation of white-box aes[C]//The 2nd International Conference on Computer Science and Its Applications. Piscataway: IEEE Press, 2009: 1-6.

[7] MULDER DE Y, ROELSE P, PRENEEL B. Cryptanalysis of the Xiao-Lai white-box AES implementation[C]//Selected Areas in Cryptography. Berlin: Springer, 2013: 34-49.

[8] LUO R, LAI X J, YOU R. A new attempt of white-box AES implementation[C]//2014 International Conference on Security, Pattern Analysis, and Cybernetics. Piscataway: IEEE Press, 2014: 423-429.

[9] KARROUMI M. Protecting white-box AES with dual ciphers[C]//Information Security and Cryptology-ICISC 2010. Berlin: Springer, 2011: 278-291.

[10] MULDER DE Y. White-box cryptography: analysis of white-box AES implementations[D]. Belgium: KU Leuven, 2014.

[11] BIRYUKOV A, BOUILLAGUET C, KHOVRATOVICH D. Cryptographic schemes based on the ASASA structure: black-box, white-box, and public-key[C]//Advances in Cryptology—ASIACRYPT 2014. Berlin: Springer, 2014: 63-84.

[12] BOGDANOV A, ISOBE T. White-box cryptography revisited: space-hard ciphers[C]//The 22nd ACM SIGSAC Conference on Computer and Communications Security. New York: ACM Press, 2015: 1058-1069.

[13] KARROUMI M. Protecting white-box AES with dual ciphers[C]//Proceedings of the 13th International Conference on Information Security and Cryptology. Berlin: Springer, 2011: 278-291.

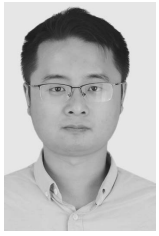
[14] XU T, LIU F, WU C. A white-box AES-like implementation based on key-dependent substitution-linear transformations[J]. Multimedia Tools and Applications, 2018, 77(14): 18117-18137.

[15] 姚思, 陈杰. SM4 算法的一种新型白盒实现[J]. 密码学报, 2020, 7(3): 358-374.
YAO S, CHEN J. A new method for white-box implementation of SM4 algorithm[J]. Journal of Cryptologic Research, 2020, 7(3): 358-374.
[16] FUKUSHIMA K, HIDANO S, KIYOMOTO S. White-box implementation of stream cipher[C]//The 3rd International Conference on Information Systems Security and Privacy. [S.n.:s.l.], 2017: 263-269.
[17] BAI K, WU C. A secure white-box SM4 implementation[J]. Security and Communications Networks, 2016, 9(10): 996-1006.
[18] BAI K P, WU C K, ZHANG Z F. Protect white-box AES to resist table composition attacks[J]. IET Information Security, 2018, 12(4): 305-313.
[19] BIRYUKOV A, CANNIERE DE C, BRAEKEN A, et al. A toolbox for cryptanalysis: linear and affine equivalence algorithms[C]// Advances in Cryptology—EUROCRYPT 2003. Berlin: Springer, 2003: 33-50.

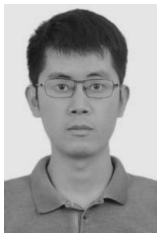
[作者简介]

王滨（1978− ），男，山东泗水人，博士，浙江大学研究员，主要研究方向为智能终端安全、物联网安全、密码学等。

陈思（1993− ），女，河南商丘人，中国电科集团 52 所工程师，主要研究方向为物联网安全、密码学及其应用。



陈加栋（1988− ），男，江苏高邮人，中国电科集团 52 所高级工程师，主要研究方向为信息安全、硬件安全等。



王星（1985− ），男，山西太原人，浙江大学在站博士后，中国电科集团 52 所高级工程师，主要研究方向为机器学习与物联网安全。

收录声明

本刊对发表的文章，拥有出版电子版、网络版版权，并拥有和其他网站交换信息的权利。本刊支付的稿酬中已经包含上述费用。

Journal on Communications has the copyright to publish electronic edition, online edition of the published articles, and has the right to exchange information with other sites. The expenses have been included in the fee paid by editorial department.

道德声明

本刊发表的论文是作者独立取得的原创性研究成果，无一稿多投；论文内容不涉及国家机密；未曾以任何形式用任何文种在国内外公开发表过；论文内容不侵犯他人著作权和其他权利。若发生一稿多投、侵权、泄密等问题，论文作者将承担全部责任。

The authors of *Journal on Communications* guarantee that their submitted articles are original and contain nothing confidential. The said article is only submitted to *Journal on Communications*. The said article has not been published before and has not been submitted elsewhere for print or electronic publication consideration. The said article is no way whatever a violation or an infringement of any existing copyright or license from the third party. Otherwise, the authors of the said article shall take the blame for the violation or infringement of the related copyright and the leakage of secrets.

通信学报

Journal on Communications

发行代号：国内2-676 / 国外M395

2021年2月25日出版　定价：98.00元

ISSN 1000-436X

02