文章编号:1001 - 9081(2007)12 - 2940 - 02

Blowfish 密码系统分析

钟黔川, 朱清新
(电子科技大学 计算机科学与工程学院,成都 610054)
(qczhong@163.com)

摘 要:Blowfish 算法自提出以后便得到了广泛应用,很多针对它的攻击也随之出现,但未见对它有实质性的挑战。针对 Blowfish 算法加密过程中出现的缺陷,给出了从 Blowfish 算法更新后得到的子密钥数组直接导出密钥数组 K 的详细过程,指出在应用中可能造成整个 Blowfish 算法被攻破。另外,用反证法证明了由于不满足前提条件,因而滑动攻击对 Blowfish 算法失效。

关键词:Blowfish;滑动攻击;分组密码
中图分类号: TP309 **文献标志码**:A

Analysis of Blowfish cryptography

ZHONG Qian-chuan, ZHU Qing-xin
(*Department of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu Sichuan* 610054, *China*)

Abstract: Since Blowfish algorithm has been proposed and applied widely, a lot of attacks aiming at it have appeared, but none of them has substantial challenge to it. Concerning the weakness of Blowfish algorithm that appeared in the encrypting process, the detailed process was given that the subkey array renewed from Blowfish algorithm could educe a key array K directly, which may result in that the whole Blowfish algorithm was broken in applications. Moreover, slide attack was proved to be not effective to Blowfish algorithm with reduction to absurdity, its precondition was not satisfied.

Key words: Blowfish; slide attacks; block cipher

Blowfish 算法是一种对称分组密码,在设计之初就充分考虑了已有对称分组密码存在的一些缺陷,特别是加强了算法抗差分分析的能力。Blowfish 算法有如下性质:1)快速,使用 32 bit 微处理器,可以达到每 18 个时钟周期加密 1 Byte 的速度;2)紧凑,运行 Blowfish 算法结构简单,易于实现,且易于确定算法强度;3)安全性可变,密钥长度可变,最长可达 448 bit,这使得用户可在安全性和速度之间进行权衡。由于它的这些特点使得它在软件加密中得到广泛应用,并在使用过程中接受了相当多的考验。

1 初始化子密钥和 S 盒

Blowfish 算法的密钥长度可在 32 bit 到 448 bit 之间变化,即字长若为 32 bit,密钥长度可为 1 Byte 到 14 Byte。这个密钥用来产生 18 个 32 bit 的子密钥和 4 个 8 × 32 的 S 盒,这些 S 盒总共有 1 024 个 32 bit 项,加上子密钥一共是 1 042 × 32 bit 或 4 168 Byte。

Blowfish 算法的密钥可表示为数组 K:

$$K_1, K_2, K_3, K_4, \cdots, K_j,\ 1 \leqslant j \leqslant 14 \tag{1}$$

子密钥可表示为数组 P:

$$P_1, P_2, P_3, P_4, \cdots, P_{18} \tag{2}$$

4 个 S 盒,每个有 256 项,每项 32 bit,可表示为:

$$\begin{aligned} &S_{1,0}, S_{1,1}, S_{1,2}, S_{1,3}, \cdots, S_{1,255}\\ &S_{2,0}, S_{2,1}, S_{2,2}, S_{2,3}, \cdots, S_{2,255}\\ &S_{3,0}, S_{3,1}, S_{3,2}, S_{3,3}, \cdots, S_{3,255}\\ &S_{4,0}, S_{4,1}, S_{4,2}, S_{4,3}, \cdots, S_{4,255} \end{aligned} \tag{3}$$

依次初始化子密钥数组 P 和 4 个 S 盒各项,方法是将常数 π 的小数部分按 32 位依次分配给这些项。对应写成十六进制为:

$$\begin{aligned} &P_1 = \text{0x243F6A88}, P_2 = \text{0x85A308D3}, \cdots,\\ &P_{18} = \text{0x8979FB1B}, S_{1,0} = \text{0xD1310BA6},\\ &S_{1,1} = \text{0x98DFB5AC}, \cdots, S_{4,255} = \text{0x3AC372E6} \end{aligned} \tag{4}$$

将 P 和 K 按位异或,并重复使用 K。对于最大长度的密钥,有:

$$P_1 = P_1 \oplus K_1, P_2 = P_2 \oplus K_2, \cdots, P_{14} = P_{14} \oplus K_{14}, P_{15} = P_{15} \oplus K_1, \cdots P_{18} = P_{18} \oplus K_4 \tag{5}$$

用当前的 P 和 S 加密 64 bit 的全 0 数据块,用输出结果替换 P_1 和 P_2。用当前的 P 和 S 加密上一步的输出结果 P_1 和 P_2,用这次的输出结果替换 P_3 和 P_4。依此类推,直到 P 和 S 全部被更新为止,每一步所使用的 Blowfish 算法都在不断变化。整个初始化 P 和 S 的过程可以描述如下:

$$\begin{aligned} &P_1, P_2 = E_{p,s}[0], P_3, P_4 = E_{p,s}[P_1][P_2]; \cdots;\\ &P_{15}, P_{16} = E_{p,s}[P_{13}][P_{14}],\\ &P_{17}, P_{18} = E_{p,s}[P_{15}][P_{16}],\\ &S_{1,0}, S_{1,1} = E_{p,s}[P_{17}][P_{18}], \cdots,\\ &S_{4,254}, S_{4,255} = E_{p,s}[S_{4,252}][S_{4,253}] \end{aligned} \tag{6}$$

其中 $E_{p,s}[Y]$ 表示使用 P 和 S 时 Blowfish 算法加密 Y 的结果。产生最终的 P 和 S 需要执行 521 次 Blowfish 算法的加密过程,所以算法不太适合于密钥经常改动的应用。此外,为了快速执行,每次使用算法时都要把 P 和 S 重算一遍,不如把 P 和 S 存起来,只需要大约 4 kByte 的存储空间,这是许多使用 Blowfish 算法加密的产品经常采用的方法。

2 Blowfish 算法加密和解密过程

加密所使用的明文被分成两半,即 LE_0 和 RE_0,各为 32

收稿日期:2007 - 06 - 22;**修回日期**:2007 - 09 - 02。

作者简介:钟黔川(1970 -),男,重庆江津人,博士研究生,主要研究方向:信息安全、混沌密码、数字水印; 朱清新(1954 -),男,四川成都人,教授,博士,博士生导师,主要研究方向:图像处理、网络与信息安全、计算机图形与视觉、最优控制理论与最优搜索。

bit。用变量 LE_i 和 RE_i 来表示第 i 轮迭代后数据的左半部分和右半部分。加密算法可用伪代码表示如下:

for i =1 to 16 do
$RE_i = LE_{i-1} \oplus P_i$;
$LE_i = F[RE_i] \oplus RE_{i-1}$;
$LE_{17} = RE_{16} \oplus P_{18}$;
$RE_{17} = LE_{16} \oplus P_{17}$;

得到的密文即变量 LE_{17} 和 RE_{17}。函数 F 的32位输入分成4 Byte,记为 a,b,c 和 d。F 可表示为:

$$F[a,b,c,d] = ((S_{1,a} + S_{2,b}) \oplus S_{3,c}) + S_{4,d} \quad (7)$$

这里的"+"为模 2^{32} 加,⊕表示按位异或。

解密过程很容易从加密算法推导出来。与加密过程对应,密文被分成 LD_0 和 RD_0,各为32 bit。用变量 LD_i 和 RD_i 来表示第 i 轮迭代后数据的左半部分和右半部分。Blowfish 算法和其他大多数分组密码一样也是把子密钥倒过来使用,与其他算法不同的是加密和解密过程顺序是一样的,而不是倒过来。解密算法可用伪代码表示如下:

for i =1 to 16 do
$RD_i = LD_{i-1} \oplus P_{19-i}$;
$LD_i = F[RD_i] \oplus RD_{i-1}$;
$LD_{17} = RD_{16} \oplus P_1$;
$RD_{17} = LD_{16} \oplus P_2$;

解密得到的明文即变量 LD_{17} 和 RD_{17}。

Blowfish 算法是传统加密算法中非常难对付的一个。Blowfish 算法的 S 盒是跟密钥有关的,不仅如此,其他一些算法,如 RC5,每轮迭代执行的函数是与数据相关的,而 Blowfish 算法的子密钥和 S 盒都是由 Blowfish 算法本身生成的。对它的密钥分析也异常困难。

3　Blowfish 算法缺陷

在本文式(6)中列出了更新 P 和 S 的过程,下面我们分析此算法的弱点。为了讨论的方便,我们将式(6)中更新 P 的式子重新写为:

$$\begin{aligned} &P_1', P_2' = E_{p,s}[0], P_3', P_4' = E_{p,s}[P_1'][P_2'], \cdots, \\ &P_{15}', P_{16}' = E_{p,s}[P_{13}'][P_{14}'], \\ &P_{17}', P_{18}' = E_{p,s}[P_{15}'][P_{16}'] \end{aligned} \quad (8)$$

用 P': $P_1', P_2', \cdots P_{13}', P_{14}', P_{15}', P_{16}', P_{17}', P_{18}'$ 表示更新后得到的子密钥数组,这里的 P 表示更新前子密钥数组,已经和 K 进行了异或,只要从 P' 导出 P:$P_1, P_2, P_3, P_4, \cdots, P_{18}$,$P$ 再与式(4)中 P 的初值按位异或可推出 K。

我们假设子密钥数组 P': $P_1', P_2', \cdots P_{13}', P_{14}', P_{15}', P_{16}', P_{17}', P_{18}'$ 已知。对于 $P_{17}', P_{18}' = E_{p,s}[P_{15}'][P_{16}']$,加密的输入值 P_{15}', P_{16}' 和输出值 P_{17}', P_{18}' 是已知的;使用的 S 盒是最初的 π 的小数部分,也是已知的;加密的子密钥数组 P 为 $P_1', P_2', \cdots, P_{13}', P_{14}', P_{15}', P_{16}', P_{17}, P_{18}$,其中只有 P_{17}, P_{18} 未知,即为我们所求的更新之前的值。由加密所用的伪代码可得:

$$\begin{aligned} &P_{17}' = LE_{17} = RE_{16} \oplus P_{18}, \\ &P_{18}' = RE_{17} = LE_{16} \oplus P_{17} \end{aligned} \quad (9)$$

这里 LE_{16} 和 RE_{16} 可由迭代等式推出,P_{17},P_{18} 也就可以求出。

对于 $P_{15}', P_{16}' = E_{p,s}[P_{13}'][P_{14}']$ 加密的输入值 P_{13}', P_{14}' 和输出值 P_{15}', P_{16}' 是已知的;使用的 S 盒是最初的 π 的小数部分,也是已知的;加密的子密钥数组 P 为 $P_1', P_2' \cdots P_{13}', P_{14}', P_{15}, P_{16}, P_{17}, P_{18}$,其中 P_{17}, P_{18} 已在上面求出,只有 P_{15}, P_{16} 未知,即为我们所求的更新之前的值。由加密所用的伪代码可得:

$$\begin{aligned} &RE_{15} = LE_{14} \oplus P_{15}, LE_{15} = F[RE_{15}] \oplus RE_{14}, \\ &RE_{16} = LE_{15} \oplus P_{16}, LE_{16} = F[RE_{16}] \oplus RE_{15}, \\ &P_{15}' = LE_{17} = RE_{16} \oplus P_{18}, P_{16}' = RE_{17} = LE_{16} \oplus P_{17} \end{aligned} \quad (10)$$

由式(10)得:

$$\begin{aligned} &P_{15} = LE_{14} \oplus P_{17} \oplus F[P_{18} \oplus P_{15}'], \\ &P_{16} = RE_{14} \oplus P_{18} \oplus P_{15}' \oplus F[P_{17} \oplus P_{16}' \oplus F[P_{18} \oplus P_{15}']] \end{aligned} \quad (11)$$

这里 LE_{14} 和 RE_{14} 可由迭代等式推出,P_{15}, P_{16} 也就可以求出。类似的可求出数组 P 的其他值。以 P':

$$\begin{aligned} &P_1' = 0x706D9FCC, P_2' = 0x1792D23A, \\ &P_3' = 0x2DB9D714, P_4 = 0x966E1439, \\ &P_5 = 0xAC21A76D, P_6 = 0x8324E988, \\ &P_7 = 0xAC0DC9DD, P_8 = 0x2C38F6B3, \\ &P_9 = 0x70619520, P_{10} = 0xFA23ECBE, \\ &P_{11} = 0x17B2F676, P_{12} = 0xEBA13A04, \\ &P_{13} = 0x8B949E61, P_{14} = 0x7A147CAF, \\ &P_{15} = 0x56CCC6B6, P_{16} = 0x4461B24D, \\ &P_{17} = 0x7361E6A1, P_{18} = 0x196A7C43 \end{aligned} \quad (12)$$

为例,求出的 P 为:

$$\begin{aligned} &P_1 = 0x243F6A88, P_2 = 0x85A308D3, \\ &P_3 = 0x13198A2E, P_4 = 0x03707344, \\ &P_5 = 0xA4093822, P_6 = 0x299F31D0, \\ &P_7 = 0x082EFA98, P_8 = 0xEC4E6C89, \\ &P_9 = 0x452821E6, P_{10} = 0x38D01377, \\ &P_{11} = 0xBE5466CF, P_{12} = 0x6C0CE934, \\ &P_{13} = 0xC0AC29B7, P_{14} = 0xC97C50DD, \\ &P_{15} = 0x3F84D5B5, P_{16} = 0xB5470917, \\ &P_{17} = 0x9216D5D9, P_{18} = 0x8979FB1B \end{aligned} \quad (13)$$

再与式(4)中 P 的初值按位异或后密钥 K 为:

$$\begin{aligned} &K_1 = 0, K_2 = 0, K_3 = 0, K_4 = 0, \cdots K_{11} = 0, K_{12} = 0, \\ &K_{13} = 0, K_{14} = 0 \end{aligned} \quad (14)$$

虽然从更新后得到的子密钥数组导出密钥数组 K 并不算攻破 Blowfish 算法,但前面我们已经说明,使用 Blowfish 算法时常将更新后的子密钥数组和 S 存起来,如放在 EEPROM、FLASH ROM 中,如果能从这些存储器获取子密钥数组和 S 盒,我们可轻易推出密钥,而根本不用改动加密/解密源程序,对应的 Blowfish 加密算法也就不起作用,轻易被攻破。同时,这一从 Blowfish 算法更新后得到的子密钥数组直接导出密钥数组 K 的过程也使我们讨论下面的问题成为可能。

4　滑动攻击 Blowfish 算法的问题

文献[2]中用滑动攻击方法对 Blowfish 算法进行了分析,认为用 2^{27} 个选择密文就可以对变形 Blowfish 算法进行攻击,从而找到与密钥有关的 S-Boxes。要对一个密码实行滑动攻击,首先,需把此密码看作同一置换 $F(x, k)$ 的乘积密码,而且 F 是弱置换,弱置换的定义是:给定 $F(x_1, k) = y_1$ 和 $F(x_2, k) = y_2$,滤出密钥 k 是容易的;其次,寻找滑动对,已知明文对 (P, C) 和 (P', C'),如果 $F(P) = P'$ 和 $F(C) = C'$,则称 (P, C) 和 (P', C') 是一滑动对;最后,利用 F 是弱置换及滑动对来滤出密钥 k。

(下转第 2944 页)

$51)/5^{-1} \bmod 42 \equiv 24$。即用户申请消息 m 的签名是：

$$Sig(12) = (P(61),24) \tag{15}$$

用户 A 得到上述签名(电子货币的某种形式)，向其他用户支付或者本身需要验证合法性，则可以进入下一阶段。

3) 电子货币的合法性验证

电子货币的接受者向某特定机构申请验证合法性，该机构利用央行所公开的盲公钥列表逐一验证下式：

$$\gamma_x\beta_i \oplus \delta\gamma = mG \tag{16}$$

当用盲公钥 $\beta_1 = P(15)$ 时，代入上式可以验证等式并不成立；当用盲公钥 $\beta_2 = P(21)$，有：

$$\gamma_x\beta_2 \oplus \delta\gamma = 51P(21) \oplus 24P(61) = P(18) \tag{17}$$

而 $mG = 12P(2) = P(18)$，因此，验证式子成立，表明该电子货币是合法银行所发行的。

4) 电子货币签发的不可否认

进行资产清算等问题，或者其他需要公开哪家商业银行发行上述电子货币时，中央银行通过验证机构提供的盲公钥 $\beta_i = P(21)$，对照商业银行所用来发行货币的公钥和商业银行名单之间的对应表即可知是银行 B_2 发行的。

3 结语

通过以上系统的实现可以发现，基于环上圆锥曲线上的群签名方案的电子货币发行系统优点比较明显，该群签名方案的安全性得到较大的提高，特别是在抵抗小加密指数和小解密指数攻击方面比经典群签名算法更加安全，能够克服经典方案的同模攻击。因此，$C_n(a,b)$ 上的 RSA 型的群签名具有应用价值。同时，与椭圆曲线上的 RSA 群签名方案相比较，体现在负元和群元整数倍计算上的快捷，使得该方案更容易实现。在算法设计中，使用了标准二进制表示来实现群元整数倍的计算，使得计算效率提高 1/4[8,9]。

参考文献：

[1] CHAUM D, Van HEYST E. Group signatures [C]// TSIOUNIS Y. Proceedings of EUROCRYPT'91, LNCS 674. Berlin: Springer-Verlag, 1991: 257 - 265.

[2] CAMENISCH J, STADLER M. Efficient group signatures for large groups [C]// Proceedings of CRYPTO'97, LNCS 1294. Berlin: Springer-Verlag, 1997: 410 - 424.

[3] CAMENISCH J, MICHELS M. Confirmer signature schemes secure against adaptive adversaries (extended abstract) [C]// Proceedings of the Advances in Cryptography. Berlin: Springer-Verlag, 2000: 243 - 258.

[4] ATENITSE G, CLANNISH J, JOYE M, *et al*. A practical and provably secure coalition - resistant group signature scheme [C]// BELLAIRE M. Advances in Cryptology-CRYPTO'00. New York: Springer-Verlag, 2000: 255 - 270.

[5] LYSYANSKAYA A, RAMZAN Z. Group blind digital signatures: A scalable solution to electronic cash [C]// Proceedings of the 2nd Financial Cryptography Conference, LNCS 1582. Berlin: Springer-Verlag, 1998: 184 - 197.

[6] 张方国，张福泰，王育民. 多银行电子现金系统[J]. 计算机学报，2001，24(5)：455 - 462.

[7] 孙琦，朱文余，王标. 环 Z_n 上圆锥曲线和公钥密码协议[J]. 四川大学学报：自然科学版，2005，42(3)：471 - 478.

[8] 王标，朱文余，孙琦. 基于剩余类环 Z_n 圆锥曲线的公钥密码体制[J]. 四川大学学报：工程科学版，2005，37(5)：110 - 116.

[9] 王标. 环 Z_n 上圆锥曲线的盲签名在电子现金中的应用[J]. 计算机应用，2006，26(1)：78 - 80.

[10] 孙琦，张起帆，彭国华. 计算群元的整数倍的一种算法及其在公钥密码体制中的应用[C]// 密码学进展：ChinaCrypt'2002. 第七届中国密码学学术会议论文集. 北京：电子工业出版社，2002：117 - 124.

[11] 杨军. 对一个密码算法的注记[J]. 四川大学学报：自然科学版，2001，38(3)，332 - 334.

(上接第 2941 页)

对变形 Blowfish 算法进行攻击主要是指假设 $P_i' = 0(i = 1,2,3,\cdots,18)$。但根据我们的计算，$P_i'(i = 1,2,3,\cdots,18)$ 根本不可能全等于 0。下面我们用反证法证明。

证明　假设 $P_i' = 0(i = 1,2,3,\cdots,18)$ 成立。用前面从更新后得到的子密钥数组直接导出密钥数组 K 的方法。

即 P'：$P_1' = 0$，$P_2' = 0$，$\cdots P_{13}' = 0$，$P_{14}' = 0$，$P_{15}' = 0$，$P_{16}' = 0$，$P_{17}' = 0$，$P_{18}' = 0$ (15)

求出的 P 为：

$P_1 = 0$，$P_2 = 0$，$\cdots P_{13} = 0$，$P_{14} = 0$，$P_{15} = 0$，$P_{16} = 0$，$P_{17} = 0$xF7FE14A5，$P_{18} = 0$x6DCCE5BC (16)

为了与 P 对应和讨论方便，我们假设 K 为 K_1，K_2，K_3，K_4，$\cdots K_j(1 \leqslant j \leqslant 18)$。然后将上式求出的 P 再与(4)式中 P 的初值按位异或后密钥 K 为：

$K_1 = 0$x243F6A88，$K_2 = 0$x85A308D3，
$K_3 = 0$x13198A2E，$K_4 = 0$x03707344，
$K_5 = 0$xA4093822，$K_6 = 0$x299F31D0，
$K_7 = 0$x082EFA98，$K_8 = 0$xEC4E6C89，
$K_9 = 0$x452821E6，$K_{10} = 0$x38D01377， (17)
$K_{11} = 0$xBE5466CF，$K_{12} = 0$x34E90C6C，
$K_{13} = 0$xC0AC29B7，$K_{14} = 0$xC97C50DD，
$K_{15} = 0$x3F84D5B5，$K_{16} = 0$xB5470917，
$K_{17} = 0$x65E8C17C，$K_{18} = 0$xE4B51EA7

很显然 $K_1 \neq K_{15}$，$K_2 \neq K_{16}$，$K_3 \neq K_{17}$，$K_4 \neq K_{18}$，这与 Blowfish 算法子密钥的产生过程相矛盾的，可知假设不成立。也就是说滑动攻击方法对 Blowfish 算法进行分析的前提条件不满足，对 Blowfish 算法的攻击是不成功的。

5 结语

提出 Blowfish 算法从更新后得到的子密钥数组可直接导出密钥数组，在应用中可能造成严重后果，但并不说明完全攻破 Blowfish 算法。我们证明了文献[2]中滑动攻击变形 Blowfish 算法的前提条件不满足。该文献中也提到文献[3]使用差分分析方法对 Blowfish 算法进行攻击。其实在文献[3]中进行差分分析也是有前提的，就是 Blowfish 算法 S 盒必须存在碰撞，文献使用概率的方法说明存在碰撞，但并没举出实证，实际查找时也是非常困难的，有待进一步研究。

参考文献：

[1] (美) STALLINGS W. 密码编码学与网络安全：原理与实践[M]. 刘玉珍，王丽娜，傅建明，译. 3 版. 北京：电子工业出版社，2004：132 - 136.

[2] BIRYUKOV A, WAGNER D. Slide attacks[C]// Fast Software EncryptionFSE'99, LNCS 1636. Berlin: Springer-Verlag, 1999: 245 - 259.

[3] VAUDENAY S. On the weak keys of blowfish[C]// Proceedings of the Third International Workshop on Fast Software Encryption, LNCS 1039. London: Springer-Verlag, 1996: 28 - 32.

[4] SCHNEIER B. Description of a new variable-length Key, 64 bit block cipher (Blowfish) [C]// Fast Software Encryption, Proceedings of the Cambridge Security Workshop, LNCS 809. London: Springer-Verlag, 1994: 191 - 204.