基于减轮 KASUMI 的 f9 算法单密钥攻击

徐新龙* 韩文报
(信息工程大学信息工程学院 郑州 450002)

摘 要：该文对 4 轮 KASUMI 的 f9 算法进行了单密钥攻击。把中间相遇攻击的思想用到 f9 算法攻击中，选取了基础密钥集与穷举密钥集，利用 K_3 与明文之间的线性关系对 f9 算法进行了中间相遇攻击，同时利用碰撞与查表技术减少了计算复杂度。最后恢复所有 128 bit 密钥需要数据复杂度是 2^{32}，优化后的计算复杂度是 $2^{125.85}$，存储复杂度是 2^{36}。

关键词：密码学；f9 算法；KASUMI 算法；中间相遇攻击
中图分类号：TN918.1 **文献标识码**：A **文章编号**：1009-5896(2013)02-0303-07
DOI: 10.3724/SP.J.1146.2012.00725

A Single Key Attack on Reduced-round KASUMI-based f9 Algorithm

Xu Xin-long Han Wen-bao
(*Institute of Information Engineering*, *Information Engineering University*, *Zhengzhou* 450002, *China*)

Abstract: An attack on f9 algorithm based on 4-round KASUMI is given with a single-key method. The meet-in-the-middle thought is applied into the attack and the based key set and the exhaustive key set are chosen. Then f9 algorithm is attacked with the linear relationship between plaintext and K_3. At the same time, collision properties and table-lookups are applied to reduce the time complexity. As a result, with 2^{32} plaintexts with the corresponding MACs, the attack needs $2^{125.85}$ f9 calculations with 2^{36} memory to recovery all the key.

Key words: Cryptography; f9 algorithm; KASUMI algorithm; Meet-in-the-middle attack

1 引言

算法 f9 是 WCDMA 安全体制中用于消息完整性保护的 MAC 算法，也叫做 3GPP-MAC 算法。f9 算法主要对 3G 无线链路中的数据进行完整性保护，防止通信中控制信令信息等在传输的过程中被恶意篡改。

f9 算法[1]于 1999 年由欧洲电信标准协会(ETSI)发布，至今已有十几年的时间，但是从公开文献中来看人们对 MAC 算法的密钥恢复攻击很少[2,3]，尤其对 f9 算法的分析仍然是空白的。文献[4]在 2003 年对 3GPP-MAC 模式进行了伪造攻击与 two-key 工作方式的密钥恢复攻击的分析，但是他的攻击没有考虑里面的分组密码性质，而只是单一的从 f9 的结构分析，给出的结果伪造攻击的数据复杂度最低在 2^{48} 量级，密钥恢复攻击的计算复杂度远远大于 2^{128}。相反，这期间倒有不少关于 f9 算法的文献尝试给出 f9 算法安全性的证明，并在基于 KASUMI 算法对一类相关密钥攻击下是安全的前提下，给出了 f9 算法的可证明安全性的证明[5,6]，以及 f9 算法的一种变形 $f9^-$ 算法可证明安全性的证明[7]。关于 f9 嵌套的 KASUMI 算法[8]，最近的分析结果有中间相遇攻击[9]，高阶差分攻击[10]，相关密钥攻击[11,12]，错误注入攻击[13]等，但都没对 KASUMI 算法构成实质上的威胁。其中最好的相关密钥攻击[12]需要 2^{26} 个选择明文，2^{33} 次 KASUMI 计算；最好的单密钥攻击[9]需要 2^{32} 个选择明文，2^{125} 次 KASUMI 计算。

本文对嵌套 4 轮 KASUMI, MAC 码长度为 64 bit 的 f9 算法，利用了单密钥攻击方法，进行了密钥恢复攻击。

2 f9 算法描述

算法 f9[1]是 KASUMI 算法的一种工作模式，该模式是在 CBC-MAC 模式基础上的改进，也称 3GPP-MAC 模式，密钥 K 长度为 128 bit。在 WCDMA 体制中 f9 的输入有计数器 COUNT-I(32 bit)，信令消息 MESSAGE(长度不定)，方向标识 DIRECTION(1 bit)及刷新器 FRESH(32 bit)，最后在完整性密钥 K 的参与下输出消息认证码 MAC 值。

计算 MAC 码时首先将输入消息级联起来并通过添“1”补“0”使得长度是 64 的整数倍，然后根

2012-06-11 收到，2012-11-12 改回
国家自然科学基金(61003291)资助课题
*通信作者：徐新龙 interestloong@163.com

据 64 bit 分组输入 f9，如图 1。其中最后一块 KASUMI 的密钥有一个密钥修改常数 KM=0$xAA\cdots AA$。

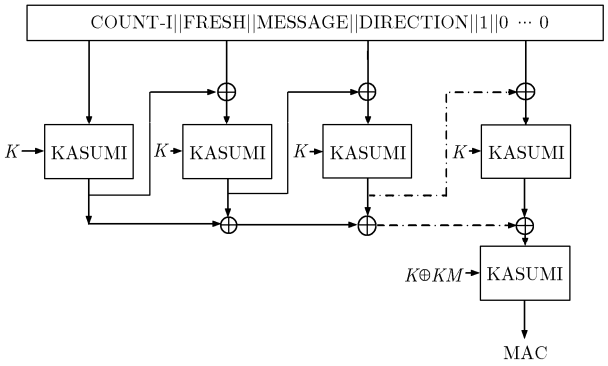

图 1 f9 结构图

本文取 m=64，即将最后一块 KASUMI 的输出 64 bit 作为 MAC 码。

3 KASUMI 描述

KASUMI[8]是一个分组长度为 64 bit 的分组密码算法，密钥长度为 128 bit。是基于 MISTY 结构的一种修改，整体为一个 Feistel 结构。算法中共有 8 轮，每轮中包括两种函数 FL 和 FO。KASUMI 的整体及局部结构见图 2。

128 bit 密钥 K，按 16 bit 分组写成 K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_8, Feistel 结构中 8 轮的子密钥扩展具体见表 1，其中 $K_i^{'}=K_i\oplus C_i$, C_i为常数，给出：C_1=0x0123, C_2=0x4567, C_3=0x89ab, C_4=0$x$cdef, C_5=0$x$fedc, C_6=0$x$ba98, C_7=0x7654, C_8=0x3210。

4 预备知识

定理 当KASUMI输入明文的左半部分 32 bit 固定，右半部分 32 bit 遍历所有 2^{32} 个值，其中必有一个明文使得第 3 轮轮函数的输入为 0x0000*ffff*。

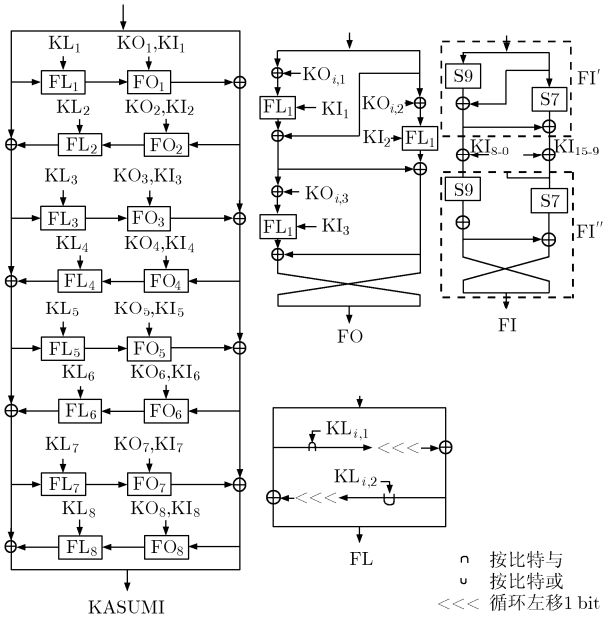

图 2 KASUMI 整体结构及各个轮函数

证明 设前 3 轮 KASUMI 的轮函数为 $F1$, $F2$, $F3$。KASUMI 的输入为 $X||Y$，其中 X 为左一半 32 bit，是固定常数，Y 是右一半 32 bit。

见图 3，第 3 轮的轮函数输入为：$A_2=X\oplus F2(F1(X)\oplus Y)$。

由于函数 FO, FL 都是可逆函数，所以轮函数 $F1$=FO(FL(···))和 $F2$=FL(FO(···))也都是可逆函数。所以当 Y 遍历 2^{32} 个值后，$F2$ 的输出也遍历了 2^{32} 个值，因此必有一个 Y 使得 A_2=0x0000*ffff*。

证毕

性质 1 当 FL 函数的输入为 0x0000*ffff* 时，不论密钥是什么，FL 函数的输出都是 0x*ffffffff*。

性质 2[9] 如图 4。当 $A_0=A_2$=0x0000*ffff* 时，B_1=0，假定除 K_3 之外的密钥{K_1,K_2,K_4,K_5,K_6,K_7, K_8}都知道，我们可以沿着第 1 轮与第 2 轮推出一个跟右半部分明文 P_r有关的值，记为 $P_r^{'}$，关系为：

表 1 KASUMI 子轮密钥扩展

轮数	$KL_{i,1}$	$KL_{i,2}$	$KO_{i,1}$	$KO_{i,2}$	$KO_{i,3}$	$KI_{i,1}$	$KI_{i,2}$	$KI_{i,3}$
1	K_1<<<1	$K_3^{'}$	K_2<<<5	K_6<<<8	K_7<<<13	$K_5^{'}$	$K_4^{'}$	$K_8^{'}$
2	K_2<<<1	$K_4^{'}$	K_3<<<5	K_7<<<8	K_8<<<13	$K_6^{'}$	$K_5^{'}$	$K_1^{'}$
3	K_3<<<1	$K_5^{'}$	K_4<<<5	K_8<<<8	K_1<<<13	$K_7^{'}$	$K_6^{'}$	$K_2^{'}$
4	K_4<<<1	$K_6^{'}$	K_5<<<5	K_1<<<8	K_2<<<13	$K_8^{'}$	$K_7^{'}$	$K_3^{'}$
5	K_5<<<1	$K_7^{'}$	K_6<<<5	K_2<<<8	K_3<<<13	$K_1^{'}$	$K_8^{'}$	$K_4^{'}$
6	K_6<<<1	$K_8^{'}$	K_7<<<5	K_3<<<8	K_4<<<13	$K_2^{'}$	$K_1^{'}$	$K_5^{'}$
7	K_7<<<1	$K_1^{'}$	K_8<<<5	K_4<<<8	K_5<<<13	$K_3^{'}$	$K_2^{'}$	$K_6^{'}$
8	K_8<<<1	$K_2^{'}$	K_1<<<5	K_5<<<8	K_6<<<13	$K_4^{'}$	$K_3^{'}$	$K_7^{'}$

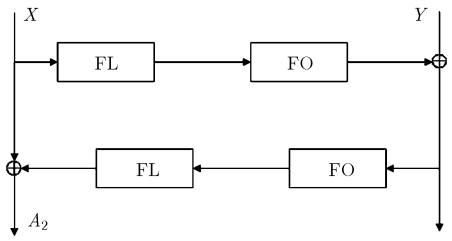

图 3 KASUMI 算法 1~2 轮

$P_r^{'} \oplus Pr = 0x(K_3 <<< 5) \,||\, 0000$，这样 K_3 就可以通过 P_r 和 $P_r^{'}$ 模二加直接得到。这个线性关系是制造中间相遇攻击重要条件。

5　4 轮 KASUMI 算法 f9 中间相遇攻击

这里考虑输入长度都为 2 块的 f9 输入(因为 f9 算法分组数 BLOCKS 最少为 2)，设为 $P_0||P_1$，其中 P_0, P_1 分别参与 f9 第 1 块与第 2 块 KASUMI 的输入，都是 64 bit, P_0 的左 32 bit 为 $0x0000ffff$, 右 32 bit 变化，P_1 是随意取的固定值。数据收集时取遍 P_0 右 32 bit 的值访问 f9 预言机，得到对应的 2^{32} 个 64 bit 的 MAC 码。

整体结构与具体节点见图 4，图 5，图 6。制造中间相遇攻击的关键是性质 2 中提到的明文跟 K_3 之间的线性关系。构造中间相遇之前，我们先给出两组密钥集，一组是基础密钥集$\{K_1, K_2, K_4^b, K_5, K_7, K_8\}$，另一组$\{K_4^e, K_6\}$是穷举密钥集(其中 K_4^b 是 K_4 中的任意 13 bit, K_4^e 是 K_4 中剩下的 3 bit)。给定一组基础密钥集，f9 中间相遇攻击的整体思路是在假设 A_0=0x0000$ffff$, A_2=0x0000$ffff$, B_1=0 的条件下，对穷举密钥集进行穷举算出 B_0, $A_1^{'}$, B_2 及第 4 轮中能算出的一些节点的值，记下它们，然后穷举 K_3，在第 4 轮已经算得那些节点的基础上继续计算，算出第 1 块 KASUMI 的输出，然后以第 1 块 KASUMI 的输出和第 2 块的 P_1 明文算出第 2 块 KASUMI 的输出，由 2 个子模块 KASUMI 的输出计算最后一块 KASUMI 第 1 轮，得到 A_9 部分比特值。另一方面，在同一组基础密钥集下，对 2^{32} 个(明文，MAC)对穷举，对最后一块 KASUMI，从 MAC 码开始，分别逆向计算出第 4 轮能算出的一些节点，按照 $P_r^{'}$ 与 P_r 的右 16 bit 相同的条件，从第 1 种方式中选出中符合条件的 K_4, K_6，同时由 K_3 的线性关系直接得到 K_3，再逆向计算两轮得到 A_9 部分比特值。由两种方式计算的 A_9 匹配进行中间相遇攻击。

由 FI 函数的结构可知，FI 函数的输出的 7~8 bit 只经过了两个 S9，所以在计算 $A_{9,l}$[7-8]时，只须对 FO 函数中前两个 FI 函数分别算两次 S9 即可(相当于两次 0.5 次 FI，即 1 次 FI 计算)，相比于计算

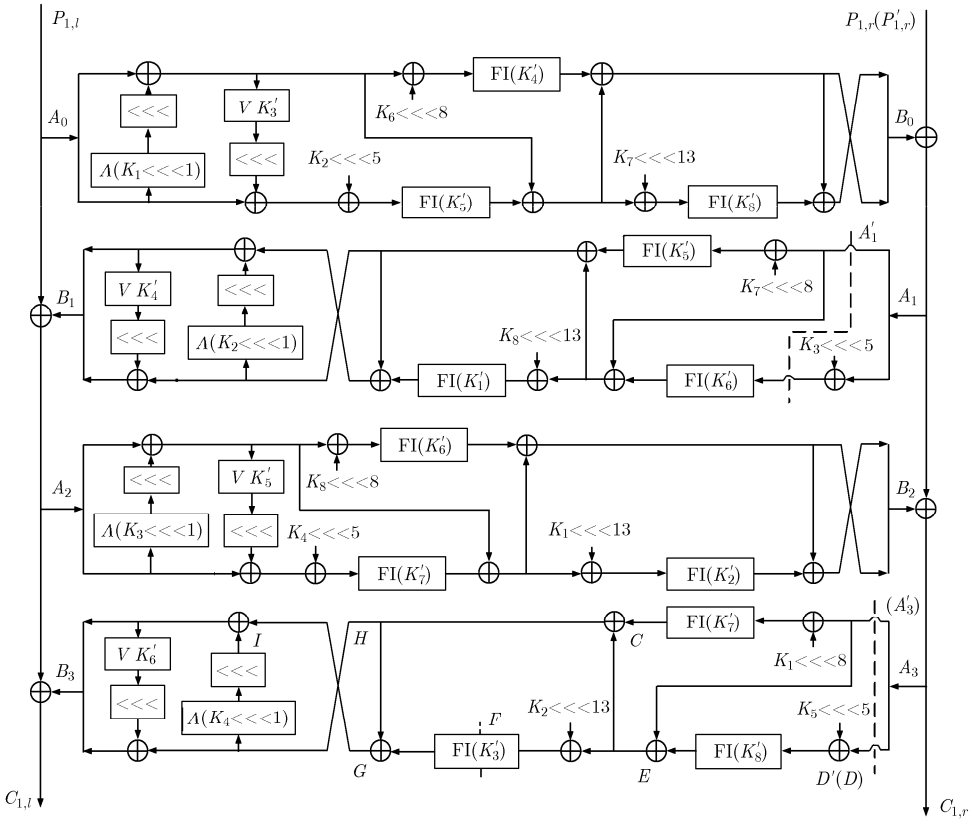

图 4 f9 第 1 块子模块

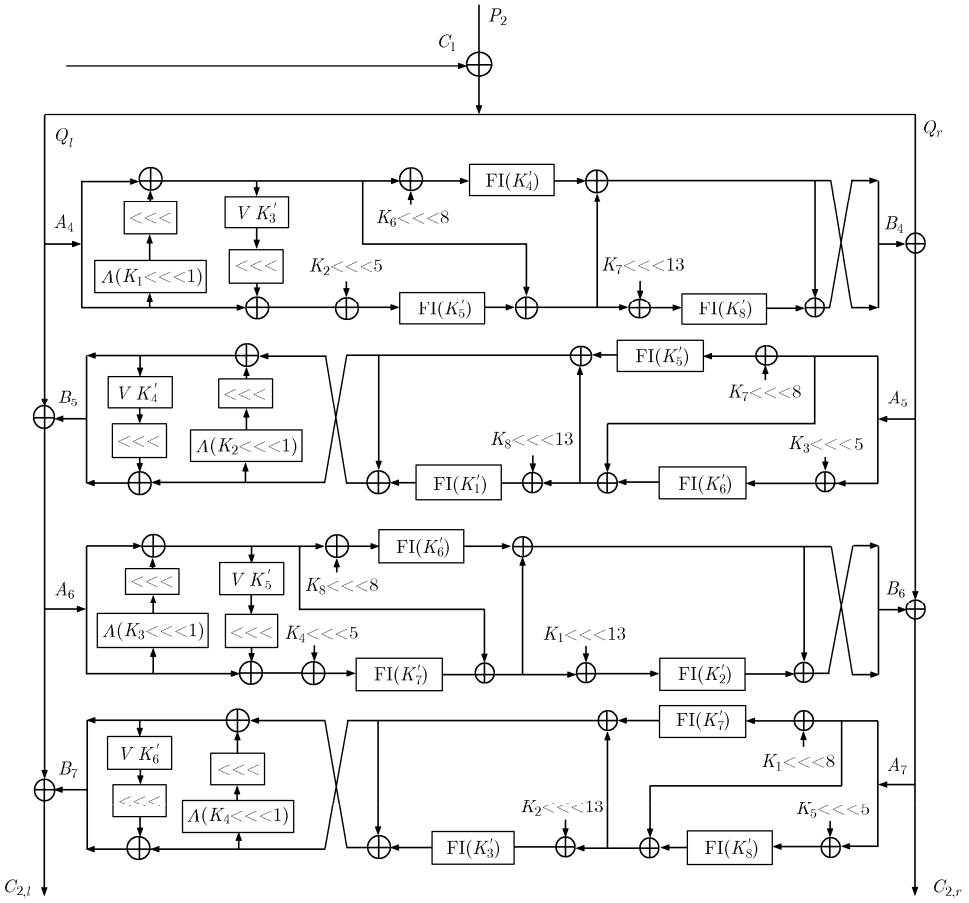

图 5 f9 第 2 块子模块

FO 函数的 3 个 FI 函数减少了 2 次 FI 计算。中间相遇攻击时，先计算 $A_{9,l}$[7-8]进行匹配提前排除掉一些错误密钥，再通过计算 A_9的其他比特进行匹配形成攻击，可以减少计算复杂度。具体做法如图 4–图 6。

符号说明：图 4，图 5，图 6 中，P_1,P_2分别对应 f9 算法输入第 1，第 2 子模块，C_1是第 1 子模块的输出，Q 是第 2 子模块 KASUMI 的输入，C_2是第 2 子模块的输出，S是最后一块 KASUMI 的输入，MAC 是最后一块的输出。A_i是每轮的轮函数输入，B_i是每轮的轮函数输出。第 1 子模块中，$A_1^{'}, A_3^{'}$是两个虚值，它们与真实值 A_1,A_3 之间的关系为：$A_1^{'} = A_1 \oplus (K_3 <<< 5 \,||\, 0x0000)$ ，$A_3^{'} = A_3 \oplus (K_3 <<< 5) \,||\, 0x0000$ 。C是第 4 轮 FO 函数中第 2 个 FI 的输出。D是第 4 轮 FO 函数在第 1 个 FI 函数输入前的值，$D' = D \oplus (K_3 <<< 5)$。$P_{1.r}^{'} = B_0 \oplus A_1^{'}$是反推出来的右半部分明文，与实际明文之间的关系为 $P_{1.r}^{'} = P_{1.r} \oplus (K_3 <<< 5) \,||\, 0x0000$ 。F是第 4 轮第 3 个 FI 函数中间加密钥之前的值(即 $F = \mathrm{FI}'(E \oplus K_2 <<< 13) >>> 7$)。最后一块中，L 是第 4 轮第 3 个FI函数中间加密钥之前的值(即 $L = \mathrm{FI}'(J \oplus MK_2 <<< 13) >>> 7$)。其它变量见图中标注。

攻击步骤：

第 1 步　收集数据。对于分组数为 2 块的输入明文，设明文为 $P_1||P_2$(P_1, P_2都是 64 bit，分别作为第 1 块与第 2 块 KASUMI 的输入明文)。其中 P_1的左 32 bit$P_{1.l}$固定为 $0x0000ffff$，右 32 bit$P_{1.r}$取遍 2^{32} 个值，P_2 是任取的一个固定值。共 2^{32} 个明文 $P_1||P_2$，通过预言机得到相应的 MAC 值，将(P_1, MAC)存入表 Table3 中。

第 2 步　预计算。由定理知，在 2^{32}个明文中必有一明文使得 A_2=$0x0000ffff$。在给定基础密钥集 $\{K_1,K_2,K_4^b,K_5,K_7,K_8\}$条件下，假设 A_2=$0x0000ffff$, B_1=0，以及 A_0=$0x0000ffff$，对穷举密钥集$\{K_4^e,K_6\}$穷举，算出 $B_0, A_1^{'}, B_2, A_3^{'}, P_{1.r}^{'}, D', C$。其中 $A_1^{'} = A_1 \oplus (K_3 <<< 5 \,||\, 0000)$ ，$P_{1.r}^{'} = P_{1.r} \oplus (K_3 <<< 5 \,||\, 0000)$ ，$A_3^{'} = A_3 \oplus (K_3 <<< 5 \,||\, 0000)$ ，$D' = D \oplus (K_3 <<< 5 \,||\, 0000)$。将(K_4^e ,K_6,$P_{1.r}$)存入表 Table1 中，并按照 $P_{1.r}$ 的值排序。将(K_4^e ,K_6, $A_3^{'}$, D', C)存入表 Table2 中，并按照 C 的值排序。

第 3 步　比较$S_{r.l}$[7-8]$\oplus B_{8.l}$[7-8]与A_{11}[7-8]$\oplus B_{10}$[7-8]是否相等。

(1)首先从正向算 $A_{9.l}$[7-8]。算法中的 $MK_i^{'} = K_i^{'} \oplus 0xAAAA$，具体见表 2。

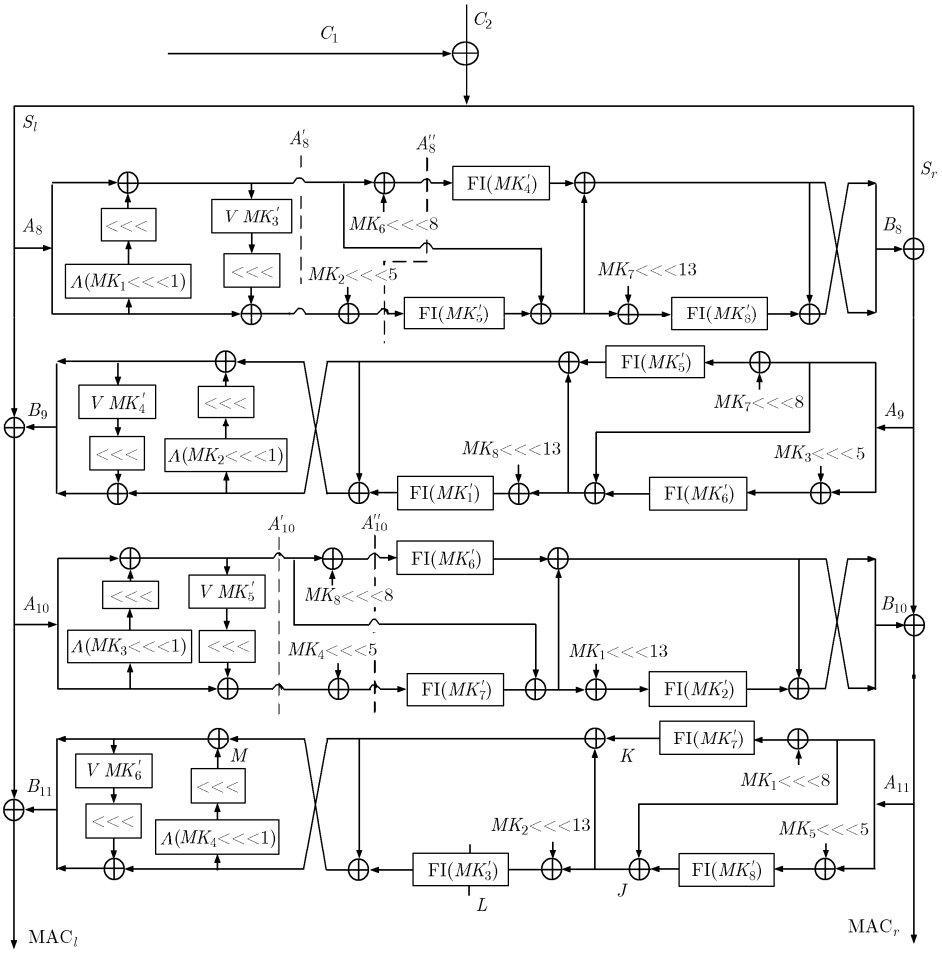

图 6 f9 最后一块 KASUMI

(2)从 MAC 值逆向计算两轮得到 $A_{9.l}$[7-8]。具体见表 3。

得到的 Table4 与 Table5 中，各有以 K_4^e,K_6,K_3 为索引的 2^{35} 个值，比较两表中 $A_{9.l}$[7-8]是否相等，排除错误密钥后，表中平均剩下 $2^{35}\times 2^{-2}=2^{33}$ 个值。

第 4 步　在剩下的 2^{33} 个值的基础上，再利用算法 1 与算法 2，最内层循环分别只需再算两次 S7 得到 $A_{9.l}$[6-0]。通过 $A_{9.l}$[6-0]的比较，排除错误密钥后，表 Table4 与 Table5 中平均剩下 $2^{33}\times 2^{-7}=2^{26}$ 个值。

第 5 步　对于剩下的 2^{26} 个值，同样利用算法 1 与算法 2，最内层循环分别只需计算两次 S7 即可，得到 $A_{9.l}$[15-9]。通过 $A_{9.l}$[15-9]的比较，排除错误密钥，表 Table4 与 Table5 中平均剩下 $2^{26}\times 2^{-7}=2^{19}$ 个值。

第 6 步　利用算法 1 与算法 2，最内层循环分别计算 1 次 FI 函数，得到 $A_{9.r}$。通过 $A_{9.r}$的比较，排除错误密钥，表 Table4 与 Table5 中平均剩下 $2^{19}\times 2^{-16}=2^3$ 个值。

第 7 步　再利用算法 1 与算法 2，分别计算 1 轮，得到 B_9的值。通过 B_9的比较，排除错误密钥，表 Table4 与 Table5 中平均剩下 $2^3\times 2^{-32}=2^{-29}$ 个值。此时，若表中还有值，则进入第 8 步，否则更换基础密钥集，从第 2 步重新开始。

第 8 步　给定的基础密钥集{K_1, K_2, K_4^e, K_5, K_7, K_8}共 93 bit，所以对给定的所有基础密钥集，表 Table4 与 Table5 中在第 7 步后共会剩下约 $2^{93}\times 2^{-29}=2^{64}$ 个值。此时通过(明文，MAC 码)对来验证密钥的正确性。因为这里 MAC 码长度为 64 bit，一次明文、MAC 对验证可以确定 64 bit，为了增大成功概率，用两组(明文，MAC 码)对验证，可筛选出正确的密钥。

计算复杂度：

因为在算法 1 第 4 步中，因为表 Table2 的大小为 2^{19}，所以同一个 C 大约有 $2^{19}\times 2^{-16}=2^3$ 个 K_4^e,K_6 与之对应。算法 2 第 3 步中，因为表 Table1 的大小为 2^{19}，所以有约 $2^{19}\times 2^{-16}=2^3$ 个 K_4^e，K_6 满足 $P'_{1.r}[15\text{-}0]=P_{1.r}[15\text{-}0]$。

第 2 步中，计算量约为 3 轮 KASUMI 算法，所

表 2 计算 $S_{r.l}$[7-8] $\oplus$ $B_{8.l}$[7-8]

算法 1：计算 $S_{r.l}[7\text{-}8] \oplus B_{8.l}[7\text{-}8]$

输入：Table0, Table2, Table6

输出：Table3

(1) For each $CC(C)$ in Table2

(2) For D from 0 to $0xffff$

(3) $E = A_{3.r}^{'} \oplus \mathrm{FI}(D, K_8^{'}), F = \mathrm{FI}'(E \oplus K_2 <<< 13)$,

$H = C \oplus E, [I = (H \wedge (K_4^b <<< 1)) <<< 1]$,

$MK_i^{'} = K_i^{'} \oplus 0xAAAA, i = 1,2,4,5,7,8$

(4) For each K_4^e, K_6 with C=CC in Table2

(5) $K_3 = (D' \oplus D) >>> 5, K_3^{'} = K_3 \oplus 0x89AB$,

$MK_3^{'} = K_3^{'} \oplus 0xAAAA, K_6^{'} = K_6 \oplus 0xBA98$,

$[I = (H \wedge (K_4^e <<< 1)) <<< 1]$,

$G = H \oplus \mathrm{FI}''[F \oplus K_3^{'} <<< 7]$,

$B_{3.r} = I \oplus G, B_{3.l} = H \oplus (B_{3.r} \vee K_6^{'}) <<< 1$,

$C_{1.l} = A_2 \oplus B_3, C_{1.r} = A_3, Q = C_1 \oplus P_2$,

$C_2 = \mathrm{KASUMI}_4(Q), S = C_1 \oplus C_2$,

$A_8^{'} = \mathrm{FL}(A_8, MK_1 <<< 1, MK_3^{'})$,

$MK_4^{'} = K_4^{'} \oplus 0xAAAA, MK_6 = K_6 \oplus 0xAAAA$,

$A_8^{''} = A_8^{'} \oplus (MK_2 <<< 5 \,||\, MK_6 <<< 8)$,

$\mathrm{Temp1} = (\mathrm{S}9((\mathrm{S}9(A_{8.r}^{''}[15\text{-}7]) \oplus 00 \,||\, A_{8.r}^{''}[7\text{-}0]) \oplus MK_4^{''}[8\text{-}0])) <<< 7$,

$\mathrm{Temp2} = (\mathrm{S}9((\mathrm{S}9(A_{8.l}^{''}[15\text{-}7]) \oplus 00 \,||\, A_{8.l}^{''}[7\text{-}0]) \oplus MK_5^{''}[8\text{-}0])) <<< 7$,

$\mathrm{Temp\_}B_{8.l} = \mathrm{Temp1} \oplus \mathrm{Temp2} \oplus A_{8.r}^{'}$,

$A_{9.l}[7\text{-}8] = S_{r.l}[7\text{-}8] \oplus \mathrm{Temp\_}B_{8.l}[7\text{-}8]$,

Store $A_{9.l}[7\text{-}8]$, Temp2, Temp1 in Table4 indexed by K_4^e, K_6, K_3,

(6) End for

(7) End for

(8)End for

表 3 计算 $A_{10.l}$[7-8] $\oplus$ $B_{10.l}$[7-8]

算法 2：计算 $A_{11.l}[7\text{-}8] \oplus B_{11.l}[7\text{-}8]$

输入：Table0, Table1, Table3

输出：Table5

(1) For $P_{1.r}$ from 0 to $0xffffffff$

(2) Search MAC from Table3.

$MK_i^{'} = K_i^{'} \oplus 0xAAAA, i = 1,2,4,5,7,8$

$J = \mathrm{FI}((\mathrm{MAC}_{r.l} \oplus MK_5 <<< 5), MK_8^{'}) \oplus \mathrm{MAC}_{r.r}$,

$K = \mathrm{FI}((\mathrm{MAC}_{r.r} \oplus MK_1 <<< 8), MK_7^{'})$,

$L = \mathrm{FI}'(J \oplus MK_2 <<< 13)$,

$[M = ((J \oplus K) \wedge (MK_4^b <<< 1)) <<< 1]$,

(3) For each K_4^e, K_6, $P_{1.r}$ in Table1, which the right 16-bits of $P_{1.r}^{'}$ has the same value with that of $P_{1.r}$

(4) K_3=$(P_{1r.l} \oplus P_{1r.l}^{'}) >>> 5, MK_3^{'} = K_3 \oplus 0xAAAA \oplus 0x89AB$,

$MK_4 = K_4 \oplus 0xAAAA, MK_6^{'} = K_6^{'} \oplus 0xAAAA$,

$[M = ((J \oplus K) \wedge (MK_4^e <<< 1)) <<< 1]$,

$B_{11.r} = M \oplus \mathrm{FI}''[L \oplus MK_3^{'} <<< 7] \oplus K \oplus J$,

$B_{11.l} = J \oplus K \oplus (B_{11.r} \vee MK_6^{'}) <<< 1$,

$A_{10}^{'} = \mathrm{FL}(B_{11} \oplus \mathrm{MAC}_l, MK_3 <<< 1, MK_5^{'})$,

$A_{10}^{''} = A_{10}^{'} \oplus (MK_4 <<< 5 \,||\, MK_8 <<< 8)$,

$\mathrm{Temp1} = (\mathrm{S9}((\mathrm{S9}(A_{10.r}^{'}[15\text{-}7]) \oplus 00 \,||\, A_{8.r}^{''}[7\text{-}0]) \oplus MK_6^{'}[8\text{-}0])) <<< 7$,

$\mathrm{Temp2} = (\mathrm{S9}((\mathrm{S9}(A_{10.l}^{''}[15\text{-}7]) \oplus 00 \,||\, A_{10.l}^{''}[7\text{-}0]) \oplus MK_6^{'}[8\text{-}0])) <<< 7$,

$\mathrm{Temp\_}B_{10.l} = \mathrm{Temp1} \oplus \mathrm{Temp2} \oplus A_{10.r}^{'}$,

$A_{9.l}[7\text{-}8] = A_{11.l}[7\text{-}8] \oplus \mathrm{Temp\_}B_{10.l}[7\text{-}8]$,

Store $A_{9.l}[7\text{-}8]$, Temp1, Temp2 in Table5 indexed by K_4^e, K_6, K_3,

(5) End for

(6)End for

以计算复杂度为 $2^{93} \times 2^{19} \times 3 = 3 \times 2^{112}$ 次单轮 KASUMI 计算，折合成 f9 计算为 2^{110} 次。

第 3 步(1)算法 1 中，第 3 步“[ ]”内的 $I = (H \wedge (K_4^b <<< 1)) <<< 1$ 与第 4 步中“[ ]”内的 $I = (H \wedge (K_4^e <<< 1)) <<< 1$ 可各看成 0.25 次 FL 计算。所以第 3 步的计算量有 1.5 次 FI 函数，0.25 次 FL 函数，循环次数平均为 $2^{93} \times 2^{16} \times 2^{16} = 2^{125}$，所以共有 $2^{128} \times 2^{-3} \times 1.5$ 次 FI, $2^{128} \times 2^{-3} \times 0.25$ 次 FL 计算。第 4 步中，有 0.5 次 FI, 0.75 次 FL，4 轮 KASUM, 1 次 FL 及 1 次 FL 计算，循环次数为 $2^{93} \times 2^{16} \times 2^{16} \times 2^3 = 2^{128}$，所以计算量共为 $2^{128} \times 13.5$ 次 FI, $2^{128} \times 5.75$ 次 FL 计算。则算法 1 共有 $2^{128} \times (13.5 + 1.5 \times 2^{-3})$次 FI, $2^{128} \times (5.75 + 0.25 \times 2^{-3})$次 FL 计算。折合成 f9 计算共 $2^{126.65}$ 次。

第 3 步(2)算法 2 中，第 2 步“[ ]”内的 $M = ((J \oplus K) \wedge (MK_4^b <<< 1)) <<< 1$ 与第 3 步中“[ ]”内的 $M = \big((J \oplus K) \wedge \big(MK_4^e <<< 1\big)\big) <<< 1$ 分别可看成 0.25 次 FL 计算。所以第 2 步的计算量为 2.5 次 FI, 0.25 次 FL，循环次数为 $2^{93} \times 2^{32} = 2^{125}$，所以共有 $2^{128} \times 2^{-3} \times 2.5$ 次 FI, $2^{128} \times 2^{-3} \times 0.25$ 次 FL 计算。第 3 步中，有 0.5 次 FI, 0.75 次 FL, 1 次 FL 及 1 次 FI，循环次数平均为 $2^{93} \times 2^{32} \times 2^3 = 2^{128}$，所以有 $2^{128} \times 1.5$ 次 FI, $2^{128} \times 1.75$ 次 FL 计算。则算法 2 共有 $2^{128} \times (1.5 + 2.5 \times 2^{-3})$次 FI, $2^{128} \times (1.75 + 0.25 \times 2^{-3})$次 FL 计算。折合成 f9 计算共 $2^{123.91}$ 次。

第 4 步及其以后的计算量可以忽略。

所以整个攻击的计算复杂度为：$2^{126.65}+2^{123.91}=2^{126.85}$ 次 f9 计算。

改进：在对 f9 的中间相遇攻击中，我们可以通过预计算和查表使得攻击速率提高。对于函数 $\mathrm{FI}(K_{i,j})$，我们可以建表 $\mathrm{Table6}[i]=\mathrm{FI}'(i)>>>7$，$\mathrm{Table7}[i]=\mathrm{FI}''(i<<<7)$，$i$=0,1,$\cdots$, 0$xffff$。则 $\mathrm{FI}(x, K_{i,j})=\mathrm{Table7}[\mathrm{Table6}[x]\oplus K_{i,j}]$。即 FI 函数可以通过 2 次查表和 1 次模二加操作来完成。为了增加算法中的 Temp1, Temp2 的计算速度，我们建表 $\mathrm{Table8}[i,j]=\mathrm{S9}(\mathrm{S9}(i)\oplus j)>>>7$，$i$=0,1,$\cdots$, 0$x1ff$, j=0,1,$\cdots$, 0$x1ff$。则算法中的 Temp1,Temp2 = (S9((S9(x) $\oplus$ 00 || y) $\oplus$ K)) <<< 7=Table8[x,00||$y\oplus K$]。即 Temp1,Temp2 的计算转化为 1 次查表和 1 次模二加。同时在算法 1，算法 2 中涉及到的循环移位操作共有 5 种，为“<<<5”，“<<<7”，“<<<8”，“<<<1”和“>>>5”。建 5 个表 Table9[i]=i<<<5, Table10[i] =i<<<7, Table11[i]=i<<<8, Table12[i] =i<<<1, Table13[i]=i>>>5, i=0,1,$\cdots$, 0$xffff$。忽略掉模二加操作，则算法 1，算法 2 的总计算量为 7.5×2^{128} 次 FL 计算，35×2^{128} 次查表，折合成 f9 计算量为 $2^{125.85}$ 次。

存储复杂度：

整个攻击过程中共有 13 个表格，它们各自的规模大小总结见表 4。

可见表中规模最大的是 Table4 和 Table5，规模都在 2^{35}，其它表的存储量相比较可以忽略，所以攻击的存储复杂度约为 2^{36}。

6 结束语

对于 f9 算法，目前还没有关于密钥恢复攻击的公开文献，本文将中间相遇攻击应用到了 4 轮 KASUMI 的 f9 算法，对 f9 算法进行了密钥恢复攻击。结果对 f9 的密钥恢复攻击在数据量 2^{32} 个选择明文及对应 MAC 码，优化后的计算复杂度为 $2^{125.85}$ 次 f9 计算及存储复杂度 2^{36} 条件下恢复了所有 128 bit 密钥。其中对于 f9 的输入块数设定为最小值，为两块 KASUMI 的情形。直觉上，这时的 f9 算法在 3gpp 模式下相当于一个 12 轮的 KASUMI 算法，相比于 KASUMI 算法要多出 4 轮，安全性明显高于 KASUMI 算法，这也是 f9 算法鲁棒性所在。本文针对了 4 轮 KASUMI 算法的 f9 算法进行了攻击，对于多轮甚至全轮 KASUMI 算法的 f9 攻击还有待于进一步的研究。

表 4 各个表的规模

Table1	Table2	Table3	Table4	Table5	Table6	$\cdots$	Table13
2^{19}	2^{19}	2^{32}	2^{35}	2^{35}	2^{16}	2^{16}	2^{16}

参 考 文 献

[1] 3GPP TS 35.201 V10.0.0. Specification of the 3GPP confidentiality and integrity algorithms; Document 1:f8 and f9 Specification[OL]. http://www.3gpp.org/ftp/Specs/html-info/35201.htm. 2011.

[2] Mitchell C J. A new key recovery attack on the ANSI retail MAC[J]. *Electronics Letters*, 2003, 39(4): 361–362.

[3] Knudsen L R. Partial key recovery attack against RMAC[J]. *Journal of Cryptology*, 2005, 18(4): 375–389.

[4] Knudsen L R and Mitchell C J. Analysis of 3gpp-MAC and two-key 3gpp-MAC[J]. *Discrete Applied Mathematics*, 2003, 128(1): 181–191.

[5] Tetsu Iwata and Tadayoshi Kohno. New security proofs for the 3GPP confidentiality and integrity algorithms[C]. Proceedings of Fast Software Encryption (FSE 2004), 2004, LNCS, 3017: 427–445.

[6] Tetsu Iwata and Kaoru Kurosawa. How to enhance the security of the 3GPP confidentiality and integrity algorithms [C]. Fast Software Encryption, 2005, LNCS, 3557: 268–283.

[7] Zhang Li-ting, Wu Wen-ling, and Wang Peng. A new variant of 3GPP-MAC with provable security and higher efficiency[J]. *Journal of Information Science and Engineering*, 2010, 26(4): 1539–1552.

[8] 3GPP TS 35.202 V10.0.0. Specification of the 3GPP confidentiality and integrity algorithms; Document 2: KASUMI Specification[OL]. www.3gpp.org/ftp/Specs/html-info/35202.htm. 2011.

[9] Jia Ke-ting, Yu Hong-bo, and Wang Xiao-yun. A meet-in-the-middle attack on the full KASUMI[OL]. http://eprint.iacr.org/2011/466.pdf. 2011.

[10] Teruo Saito. A single-key attack on 6-round KASUMI[OL]. www.eprint.iacr.org/2011/584. pdf. 2011.

[11] Nguyen P H, Robshaw M J B, and Wang Hua-xiong. On related-key attacks and KASUMI: the case of A5/3[C]. Progress in Cryptology-Indocrypt, 2011, LNCS, 7107: 146–159.

[12] Dunkelman O, Keller N, and Shamir A. A pratical-time attack on the A5/3 cryptosystem used in third generation GSM[C]. CRYPTO 2010, LNCS, 6223: 393–410.

[13] Lee Yuseop, Sung Jaechul, and Hong Seokhie. Fault injection attack on A5/3[C]. Proceedings in ISPA 2011, Busan, May 26–28, 2011: 300–303.

徐新龙： 男，1988 年生，硕士生，研究方向为分组密码的分析与应用.

韩文报： 男，1963 年生，教授，研究方向为信息安全.