文章编号:1007 - 2985(2012)05 - 0066 - 04

MyriaNed 传感器平台分组密码算法的实现与优化*

温雅敏[1],何晓炼[2]

(1.广东商学院数学与计算科学学院,广东 广州 510320;2.华南师范大学计算机学院,广东 广州 510631)

摘 要:在 MyriaNed 传感器平台上,采用面向软件的设计思路,很好地平衡了算法的安全性和软硬件实现上的效率与开销,从而实现了 AES,SKIPJACK 及 KLEIN 这 3 种分组密码算法.通过面向软件的设计思路,很好地平衡了 KLEIN 算法的安全性和加密速度.比较在 MyriaNed 平台消息分组加密速度,KLEIN 算法相对于 AES 算法的速度优势明显.

关键词:传感器网络;MyriaNed;分组密码算法

中图分类号:TP309.7 **文献标志码**:A **DOI**:10.3969/j.issn.1007 - 2985.2012.05.017

集成了传感器、微机电系统和网络 3 大技术而形成的传感器网络是一种全新的信息获取和处理技术.在医疗健康领域,如果在住院病人身上安装特殊用途的传感器节点,如心率和血压监测设备,利用传感器网络,医生就可以实时了解病人的病情,进行及时处理.文中所介绍的 MyriaNed 传感器平台[1-2]是荷兰 ALWEN 项目组[3]开发中拟用医疗传感器网络的无线传感器节点,具备无线传感器网络低功耗、易开发和易维护的特点.

由于医疗信息的保密性和隐私性,在医疗传感器网络传输、处理信息必须使用密码学手段进行保护[4-5].分组密码算法作为密码学方案的基础构件,能进一步构造哈希函数和消息认证码,同时其软硬件开销在传感器平台也比较合适.分组密码具有其独特优点:明文信息良好的扩展性和插入的敏感性,不需要密钥同步,较强的适用性,适合作为加密标准.但是也有其缺点:加密速度慢,错误扩散和传播[6-7].对于需要运用在无线传感器网络中的加密算法,必须在算法速度及软硬件开销上适合自己的平台,因此需要对将要移植到 MyriaNed 传感器平台的已公开发表的分组加密算法进行算法优化,以满足平台要求.

笔者介绍并实现了 AES,SKIPJAC 和 KLEIN 这 3 种分组密码算法,通过比较其内存开销和加密速度等,观察各密码算法在平台上的表现,然后对 KLEIN 算法进行一些软件上优化,将 GF(2^8)上的乘 2 用 1 次左移运算和 1 次条件异或运耳表示,采用面向软件的设计思路,很好地平衡了算法的安全性和软硬件实现上的效率与开销.

1 适用于 MyriaNed 平台的轻量级分组密码

1.1 AES 密码算法

AES(The Advanced Encryption Standard)是美国国家标准与技术研究所用于加密电子数据的规范,它被预期能成为人们公认的加密包括金融、电信和政府数字信息的方法.AES 算法的核心由 3 部分组成:1 个初始的轮密钥;$Nr-1$ 轮轮变换;1 个结尾轮变换.AES 中各个不同的变换都是在称之为状态的中间结果上进行的.状态是中间的密码结果,它可以用字节的 1 个矩阵来表示,该矩阵有 4 行,列数为 Nb.Nb 等于分组的长度除以 32,所以在 128 位分组长度的 AES 中,$Nb=4$.AES 使用了轮变换,但与许多密码算法不同的是:随着密钥长度的不同,AES 轮数(Nr)也是有变化的.128 位密钥长度对应 10 轮,192 位密钥长度对应 12 轮,256 位密钥轮数对应 14 轮,文中采用 128 位密钥长度 AES 算法.

* **收稿日期**:2012 - 07 - 11

基金项目:广东商学院博士科研基金资助项目(11BS41301)

作者简介:温雅敏(1981 -),女,江西赣州人,广东商学院数学与计算科学学院讲师,博士,主要从事密码学研究

在 AES 算法核心开始处理前,算法将输入的明文分成 16 个 8 位的分组,按顺序依次映射到状态矩阵中,算法核心运算完成之后,将状态矩阵重新排列成 1 维数组,即可得到密文分组. 下列代码为 MyriaNed 平台已实现 AES 所采用的算法过程.

```
#define AES-128 Encryption (message,key) {
for(int i = 0;i < 10;i++) {
    state = AddRoundKey(sessage,key);
    state = SubBytes(state);
    state = ShiftRows(state); state = MixColumns(state);
    key = UpdateKey(key);
   }
   }
```

1.2 SKIPJACK 密码算法

1994 年美国政府公布了密钥托管加密标准,在密钥托管加密标准中主要使用了防篡改的密码芯片 CLIPPER,在该芯片中使用了分组码密码算法 SKIPJACK. 该算法具有适用于嵌入式环境下软硬件实现的优势,得到了学术界与工业界的广泛认可. SKIPJACK 加密算法采用了 80 bit 密钥把 64 bit 输入分组转换成 64 bit 输出分组,算法采用的是单密钥,通过交替地使用如下 2 个规则(A 和 B)来对 8 个字节数据分组进行加密,算法实现的主要过程如下所示.

(1) 定义规则 A:

```
#define RULE_A(skey,w1,w2,w3,w4,counter,tmp,bLeft,bRight ) {
tmp = w4;
    w4 = w3;
    w3 = w2;
    w2 = G(skey,w1,bLeft,bRight);
    w1 = w2 ^tmp ^ counter;
counter++;}
```

(2) 定义规则 B:

```
#define RULE_B(skey,w1,w2,w3,w4,counter,tmp,bLeft,bRight ) {
tmp = w1;
    w1 = w4;
    w4 = w3;
    w3 =tmp ^ w2 ^ counter;
    w2 = G(skey,tmp,bLeft,bRight);
counter++;}
```

(3) G 变换(其中 F 即为定义的 F-table):

```
#define G(key,b,bLeft,bRight)
    (bLeft   = (b >> 8),
bRight  = b,
bLeft  ^= pgm_read_byte(&F[bRight ^ (key)[0]]),
bRight ^= pgm_read_byte(&F[bLeft  ^ (key)[1]]),
bLeft  ^= pgm_read_byte(&F[bRight ^ (key)[2]]),
bRight ^= pgm_read_byte(&F[bLeft  ^ (key)[3]]),
       (bLeft << 8) | bRight)
```

(4) 加密过程:输入为 W,$1\leqslant i\leqslant 4$,counter 置为 1. 执行 8 步规则 A,然后转换到 8 步的规则 B,再重新回到规则 A 进行接下来的 8 步,最后完成加密在规则 B. counter 每步自增 1,输出为 Wi^{32},$1\leqslant i\leqslant 4$.

```
voidskipjack_encryptTask(uint8_t *ekey,uint8_t *in,uint8_t *out)
{
    uint8_t counter = 1,bLeft,bRight;
```

```
    uint8_t * skey = ekey;
    uint16_t w1,w2,w3,w4,tmp;
    w1 = (uint16_t)in[0]<<8 ^ (uint16_t)in[1];
    w2 = (uint16_t)in[2]<<8 ^ (uint16_t)in[3];
    w3 = (uint16_t)in[4]<<8 ^ (uint16_t)in[5];
    w4 = (uint16_t)in[6]<<8 ^ (uint16_t)in[7];
    while (counter <= 8) {
      RULE_A(skey,w1,w2,w3,w4,counter,tmp,bLeft,bRight );
    skey += 4;
    }
    skey = ekey+2;
    while (counter <= 16) {
      RULE_B(skey,w1,w2,w3,w4,counter,tmp,bLeft,bRight );
    skey += 4;
    }
    skey = ekey+4;
    while (counter <= 24) {
      RULE_A(skey,w1,w2,w3,w4,counter,tmp,bLeft,bRight );
    skey += 4;
    }
    skey = ekey+6;
    while (counter <= 32) {
      RULE_B(skey,w1,w2,w3,w4,counter,tmp,bLeft,bRight );
    skey += 4;
    }
    out[0] = (uint8_t)(w1 >> 8);out[1] = (uint8_t)w1;
    out[2] = (uint8_t)(w2 >> 8);out[3] = (uint8_t)w2;
    out[4] = (uint8_t)(w3 >> 8);out[5] = (uint8_t)w3;
    out[6] = (uint8_t)(w4 >> 8);out[7] = (uint8_t)w4;
}
```

1.3 Klein 密码算法

Klein 算法是一种面向软件实现的轻量级分组密码算法,固定 64 bit 的分组长度.算法密钥长度可选择 64,80,96 bit,但同时需要对应不同的迭代轮数(即 12、16 和 20 轮).Klein 算法尽可能地采用面向字节的处理模式,以实现受限环境的高效性.在算法的非线性模块,Klein 算法采用了具有自反性质的 4 bit S 盒,使得算法仅需要付出 1 个 S 盒的代价来实现加密解密运算.在扩散模块,Klein 将 AES 的 MixColumns 函数变形为算法中的 MixNibbles 函数,同时与面向字节的循环左移函数 RotateNibbles 相结合,这样保证了算法的软硬件效率.在密钥调度模块,Klein 选择了比较复杂的处理方法,从而保证基于 Klein 的哈希函数也具有较高的安全性,KLEIN 算法的各步骤优化采用了文献[6]中的优化方法.

2 各算法执行结果比较

在 MyriaNed 平台的调试和测试结果并不像一般的软件开发调试过程,不能直接在控制台输出想要的结果,这里需要用到与 MyriaNed 平台配套的图形界面串口调试终端,其测试加密算法时间开销代码如下:

```
startTime = mcGetUptime();      //执行加密前先获取当前时间;
/*执行加密*/
for(i = 0;i < EXPERIMENT_TIMES;i++)
  {
klein64_encrypt(message,key,cipher);  //四种算法分别测试执行
      //kleinImp64_encrypt(message,key,cipher);
```

```
        //aes(message,cipher,key);
        //skipjack_encryptTask(ekey,message,cipher);
    }
endTime = mcGetUptime();  //加密完成再次获取当前时间;
mnUartPutStr("n The encryption time is ");  //用于在终端输出;
mnUartPutDec(endTime -startTime);     //结束时间-起始时间即为算法执行时间
  /*密文输出*/
mnUartPutStr("n The ciphertext is ");
  for (i = 0;i < BLOCK_LENGTH;i++) {
        mnUartPutHex2(cipher[i]);
  }
```

先将编译过后的测试程序文件写进 MyriaNed 平台，然后把 USB 控制权交回给系统，运行串口调试终端. 设置好各个参数，便能在终端调试程序并观察输出，各算法时间开销如表 1 所示.

表 1　各算法在 MyriaNed 平台执行速度比较

算法	明文	密文	加密速度/ms
AES		c2cd4fa92e39f05d	2 047
KLEIN(LUT)	0x01,0x02,0x03,0x04,	33f631e78483194d	967
KLEIN(Xtime)	0x05,0x06,0x07,0x08	33f631e78483194d	835
SKIPJACK		20ce1519493a1386	530

从实验结果可以看出，KLEIN 算法相对于 AES 算法，具有更好的低功效性，内存开销优势明显. 从表 1 中数据可以看出，KLEIN 加密速度提升明显. 用 xtime 函数实现算法中 MixNibbles 步骤比查表方法实现又更具优势，内存开销进一步降低，算法速度也有一定的提升. 从加密速度考虑，SKIPJACK 算法在 MyriaNed 平台表现出色，速度优势明显.

3　结语

阐述了无线传感器网络及 MyriaNed 传感器平台，并在该平台实现了当前已经公开发表的分组密码算法，并对已经实现的分组密码算法进行优化，使其在软硬件开销上更能够适应 MyriaNed 平台.

参考文献：

[1]　周贤伟. 无线传感器网络与安全 [M]. 北京：国防工业出版社，2007.
[2]　任丰原，黄海宁，林　闯. 无线传感器网络 [J]. 软件学报，2003，14(7)：1 282－1 291.
[3]　孟庆树，王丽娜. 密码编码学与网络安全 [M]. 北京：电子工业出版社，2006.
[4]　贾春福，刘春波. 计算机安全原理与实践 [M]. 北京：机械工业出版社，2008.
[5]　戎小芳. 高级加密标准(AES)算法研究及实现技术[J]. 科技情报开发与经济，2007(30)：233－234.
[6]　CHESS. MyriaNed SDK User Guide [EB/OL]. [2012－07－11]. http://www.chess.nl.

Optimized Implementation of Block Cipher Algorithms on MyriaNed Sensor Platform

WEN Ya-min[1], HE Xiao-lian[2]

(1. School of Mathematics and Computational Science, Guangdong University of Business Studies, Guangzhou 510320, China;
2. School of Computer Science, South China Normal University, Guangzhou 510631, China)

Abstract: In this paper, we mainly introduce a software-oriented method to implement AES, SKIPJACK and KLEIN block ciphers on MyriaNed platforms. The implementation is well-balanced by considering security, performance and costs. Based on the comparison of memory costs and processing speed, KLEIN shows obvious better results than AES on MyriaNed Platform.

Key words: sensor net; MyriaNed; block cipher algorithms　　（责任编辑　陈炳权）