基于字的流密码的分布式解密

刘志高[①②] 张福泰[①] 徐 倩[①]
[①](南京师范大学数学与计算机科学学院 南京 210097)
[②](安徽工业大学职业技术学院 马鞍山 243011)

摘 要 该文分析了 Magnus Öberg 提出的简单加法流密码的分布式解密方案，指出了其最主要的缺点是：加解密要在不同的有限域内进行、加解密运算所依赖的域的阶必须为素数而且要满足一定的关系。提出了基于字的流密码的分布式解密方案。新提出的方案有两个主要优点：一是将加解密统一在同一个有限域内进行，且域的阶不再要求是素数；二是在安全性和效率两方面比原有方案均有了明显的提高。在应用方面，新方案可应用于诸如数据库信息的保护，Ad-hoc 网中分布式密钥管理，等等。

关键词 流密码，分布式解密，秘密分享，密钥流
中图分类号：TN309.7 **文献标识码**：A **文章编号**：1009-5896(2006)07-1312-05

Distributed Decryption of Word-Oriented Stream Ciphers

Liu Zhi-gao[①②] Zhang Fu-tai[①] Xu Qian[①]
[①](*School of Mathematics and Computer Science, Nanjing Normal University, Nanjing* 210097, *China*)
[②](*College of Vocational Technology, Anhui University of Technology, Ma'anshan* 243011, *China*)

Abstract The distributed decryption scheme for simple addition stream ciphers proposed by Magnus Öberg is analyzed. Its main disadvantages, i.e. the operations of encryption and decryption are implemented in different finite fields, and the orders of the two fields must be primes satisfying a certain relation, are pointed out. A new distributed decryption scheme for word-oriented stream ciphers is presented. Compared with Magnus Öberg's scheme, the proposed scheme has two main advantages. Firstly, in the new scheme, the encryption and decryption operations are implemented in the same finite field, and the size of the field is not required to be a prime. Secondly, the security and efficiency of the new scheme are greatly improved. For applications, the new scheme can be applied in the protection of data confidentiality in a database, distributed key management in Ad-hoc networks, etc.

Key words Stream cipher, Distributed decryption, Secret sharing, Key stream

1 引言

计算机网络的快速普及与应用，使得网络与信息安全受到了越来越多的关注。利用密码算法对机密数据和信息进行加密保护乃是网络与信息安全中的重要方法。在许多场合，都需要对一些密码算法实行分布式解密。例如，某公司要将其机密信息存储在数据库中，只允许授权用户能访问数据库，而非授权用户不能读，也不能修改数据库中的信息。传统的解决方案是对数据库信息进行加密保护，用户通过终端向服务器发送请求访问数据库，服务器对请求作检查(如采用身份认证等一系列访问控制技术)，确认是合法请求时，再向数据库调取机密信息(密文形式)，解密后发送或直接发送给终端(由终端解密)。该方案的系统结构如图 1 所示。

图 1 传统解密方案

Fig.1 Traditional decryption scheme

显然，在传统的解决方案中，对机密信息的保护程度依赖于如下 3 个方面：一是所选用的加密算法的安全强度；二是解密密钥如何处理；三是解密在哪里进行。在传统方案中，解密是由单个机器完成的，无论是在服务器还是在终端进行解密，解密密钥势必会泄露给该机器，这无疑给攻击者留下了机会。

如果能利用门限密码技术(如Shamir秘密分享方案[1])将解密密钥在一组机器间进行秘密分享，而且将解密操作由一组机器来共同合作完成，使参于解密的任一机器都不知道解密密钥、密文及明文，则攻击者很难有所作为。这就是分布式解密的思想。Magnus Öberg在文献[2]中详尽分析了几种方

2004-11-15 收到, 2005-07-27 改回
江苏省高校自然科学研究计划重点项目(03KJA520066)和教育部网络与信息安全重点实验室(西安电子科技大学)资助课题

案，最后得出的结论是分布式解密方案为最理想，可以很好地防范一些攻击(如窃听、拒绝服务攻击等)。图 2 显示了分布式解密方案的系统结构。

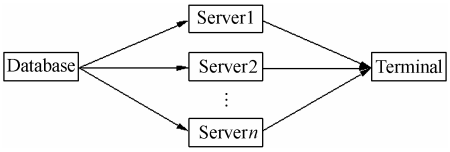

图 2　分布式解密方案

Fig.2　Distributed decryption scheme

然而，Magnus Öberg 在文献[2]中提出的方案是一种简单加法流密码的分布式解密方案，仍存在一些不足之处。为克服这些缺点，我们提出了一种新的方案——基于字的流密码的分布式解密方案。

本文将在第 2 节简述 Magnus Öberg 方案，并详细分析其缺点；在第 3 节给出一些预备知识之后；第 4 节将详细介绍基于字的流密码的分布式解密方案；第 5 节给出一个实例；第 6 节对该方案的安全性和效率作了简要分析；第 7 节作出了结论并探讨了未来要做的工作。

2　简单加法流密码的分布式解密方案及其缺点

为便于分析，现对简单加法流密码的分布式解密方案简述如下。

用简单加法流密码体制将数据库加密，加密算法为：$c_i=m_i+k_i$。其中m_i为机密信息明文的二进制位，k_i为密钥流生成器产生的密钥流的第i位子密钥，加法为模 2 加法运算。利用(t，n)Shamir秘密分享方案来实现对解密密钥及密文的分享。具体如下：选择素数$q>n$，对每一个k_i，在GF(q)上随机选取$t-1$次多项式：

$$f_i(x)= k_i+ a_i^1x+a_i^2x^2+\cdots+a_i^{t-1}x^{t-1}$$

其中$a_i^j(j=1,2,\cdots,t-1)$是在GF(q)中随机选择的元素。将$f_i(j)$的值(即为子密钥k_i的份额)存储在第j个服务器上($j=1,2,\cdots,n$)。当确认有合法请求访问数据库时，数据库将为密文的每一个二进制位c_i，在同一GF(q)上随机创建多项式：

$$g_i(x)= c_i+ b_i^1x+b_i^2x^2+\cdots+b_i^{t-1}x^{t-1}$$

其中b_i^j ($j=1,2,\cdots,t-1$)是在GF(q)中随机选择的元素。然后将$g_i(j)$的值(即密文c_i的份额)发送到第j个服务器上($j=1,2,\cdots,n$)。

文献[2]已经证明：在第j个服务器上只需计算$g_i(j)$, $f_i(j)$即可得伪明文份额 $m_i'(j)$。然后在终端只需要t个伪明文份额，再利用拉格朗日插值公式即可恢复出伪明文的第i位 m_i'，而它可唯一映射成明文的第i位m_i ，这样便实现了分布式解密。

注意：以上的加法及乘法运算均是模 q 加法及模 q 乘法运算。

不难看出，上述方案中，加密是在 GF(2)上进行的，而解密是在 GF(q)上进行的。由于使用了 Shamir 秘密分享方案，因此，对 GF(q)的选择不能是任意的，它必须满足如下两个条件：(1) GF(q)中的元素个数至少应与可能的秘密的个数一样多；(2) 对于(t，n)门限方案来说，还必须要求 $q>n$。这是因为不同的分享者(Shareholder)必须与 GF(q)上的不同的非零元素相对应。 而一般情况下，$n>2$，从而有 $q>2$。这样必然会造成加、解密在不同的有限域中进行。正是因为这一点，上述方案中解密直接得到的不是明文而是伪明文，为保证能将伪明文唯一映射成明文，文献[2]证明了 q 的选择还必须满足如下第 3 个条件：$q>2p-2$ 且 p，q 均为素数。其中加密是在 GF(p)中进行而解密是在 GF(q)中进行。证明详见文献[2]的引理 6.1。

经过认真分析，我们发现文献[2]中所提出的简单加法流密码的分布式解密方案存在如下 4 个缺点：

(1) 加、解密不是在同一有限域中进行的。加密操作及秘密分享都是按位进行的(即一次只能处理 1bit 字符)，效率低下；

(2) 对 p，q 有较强的限制，即要求 p，q 均为素数且 $q>2p-2$；

(3) 利用秘密分享方案的恢复算法不能直接恢复出明文而只能恢复出伪明文， 伪明文还需要经过映射变换才能转化为明文；

(4) 其方法不具有广泛的适用性，尤其不适用于对目前被认为是实用的流密码算法，如RC4[3]，SNOW[4]等，进行分布式解密。其原因是，目前被认为是安全高效的流密码算法都是基于字的。在字长为w的流密码体制中，加密是在GF(2^w)上进行的，显然有$p=2^w$不是素数，并且解密及密钥流生成器中的操作也都是在GF(2^w)上进行的。因此，在进行秘密分享时，若像文献[2]那样一次只对 1bit字符进行秘密分享，显然是不匹配、不同步的。

3　预备知识

3.1　基于字的加法流密码简述

现行流密码都趋向于基于字的加法流密码。如：RC4，SNOW 等。密钥流生成器一般由驱动部分和非线性变换部分组成。驱动部分一般由基于字的线性反馈移位寄存器(Linear Feedback Shift Register，LFSR)组成。由 LFSR 产生周期很长的输入字序列(即驱动序列)，将此输入字序列输入到非线性变换部分，经非线性变换后产生的输出字序列即为用于加密的密钥字序列。加密是将密钥字序列与明文字序列中对应的字进行按位“异或”运算来得到密文字序列中相应的字。每次处理都是以字为单位的。其中每个字均是由 w(w 为字长)位比特串组成。

例如：对于字长为w的加法流密码来说，设其密钥流生成器产生的密钥字序列为$k_1k_2k_3k_4\cdots$，明文字序列为$m_1m_2m_3m_4\cdots$，密文字序列为$c_1c_2c_3c_4\cdots$。其中 $k_i=k_i^{(w-1)}\ k_i^{(w-2)}\cdots$

$k_i^{(0)}$，$m_i = m_i^{(w-1)}\ m_i^{(w-2)}\cdots\ m_i^{(0)}$，$c_i = c_i^{(w-1)}\ c_i^{(w-2)}\cdots c_i^{(0)}$，$k_i, m_i, c_i \in \{0, 1\}^w$ (i=1,2,3,…) 而$k_i^{(j)}$，$m_i^{(j)}$，$c_i^{(j)} \in \{0, 1\}$(j=0,1,2,…,$w-1$; i=1,2,3,…)。则密文字$c_i = m_i + k_i = c_i^{(w-1)}\ c_i^{(w-2)}\ \cdots\ c_i^{(0)}$。其中$c_i^{(j)} = k_i^{(j)} + m_i^{(j)} \pmod 2$，($j$=0,1,2,…,$w-1$; i=1,2,3,…)。

3.2 GF(2^w)上的Shamir秘密分享方案

一个秘密分享方案涉及两方参与者，一方是拥有秘密并分发秘密份额的分发者，另一方是共同分享秘密的一组分享者。Shamir(t，n)秘密分享方案以有限域上的拉格朗日多项式插值法为基础的。涉及一个分发者D和n个分享者P_1，P_2，…，P_n。包含两个算法：一个是份额分配算法，另一个是秘密的恢复算法。假定要分享的秘密s是从有限域GF(q)中随机选取的，其中q是素数或素数的幂。份额分配算法由分发者执行，首先他随机选取GF(q)上的$t-1$ 次多项式$f(x)=s+a_1x+a_2x^2+\cdots+a_{t-1}x^{t-1}$，其中$s$是要分享的秘密，系数$a_i$是从GF($q$)中随机选择的，$a_{t-1}\neq 0$。然后分发者计算$f(1)$，$f(2)$，…，$f(n)$，并通过秘密信道把它们依次发送给$P_1$，$P_2$ …，P_n。$f(j)$就是P_j所持有的秘密份额(j=1,2,…,n)。秘密的恢复算法由n个分享者执行。至少t个分享者合作，利用他们所持有的份额，按照拉格朗日多项式插值法才可恢复出被分享的秘密s。少于t个分享者合作，不仅无法恢复被分享的秘密，而且得不到关于秘密s的任何信息。

在本文使用的秘密分享方案中，将取$q=2^w$。计算秘密份额，以及恢复秘密时，所有的运算都在有限域GF(2^w)中进行。

注：GF(2^w)上的加法及乘法运算

我们采用GF(2^w)中元素的多项式表示法。把GF(2^w)看成GF(2)[x]/($m(x)$)，其中$m(x)$是GF(2)上的一个w次不可约多项式。GF(2^w)中的一个元素表示为GF(2)上的一个次数不超过$w-1$ 的多项式$b_{w-1}x^{w-1} + b_{w-2}x^{w-2} + \cdots + b_1x + b_0$，也可简写为一个比特串$b_{w-1}\ b_{w-2}\cdots\ b_1\ b_0$。在上述多项式表示中，GF($2^w$)中的两个元素的和仍是一个次数不超过$w-1$ 的多项式，其系数是原来两个元素对应系数的模 2 加(比特“异或”)。GF(2^w)上的两个元素的乘积就是这两个元素对应的多项式在模$m(x)$下的积。例如：w=8 时，可先确定一个GF(2)上的 8 次不可约多项式，如$m(x)=x^8+x^4+x^3+x+1$(参见文献[5])，则GF(8)上的两个元素“57”(十六进制表示)与“83”的乘积可表示为$(x^6+x^4+x^2+x+1)*(x^7+x+1)=x^7+x^6+1 \pmod{m(x)}$，即二者的乘积为“$c1$”。

4 基于字的流密码的分布式解密

如前所述，基于字的流密码的加、解密操作是在GF(2^w)上进行的。一般情况下，w=8,16,32 等。故利用(t，n)门限方案进行秘密分享时，一般均有$n<2^w$。因此，可选择GF(2^w)上的插值多项式来进行秘密分享。这样做的好处就是可使加解密在同一个有限域内进行。

设由基于字的密钥流生成器产生的密钥字序列$K=k_1k_2k_3k_4\cdots$，及由它加密的密文字序列$C=c_1c_2c_3c_4\cdots$(其中k_j及c_j均是GF(2^w)中的元素，即是长为w的比特串)。现在我们给出在同一个群体中分享密钥流K和密文流C的方法。

密钥流与密文流的分享 在GF(2^w)上利用Shamir (t，n)门限方案。分发者为D，n个分享者为P_1，P_2，…，P_n。设α_1，α_2，…，α_n是GF(2^w)中n个不同的非零元素(依次对应于n个分享者)。对密钥流的每一个字k_i，密文流的每一个字c_i，分发者(加密方)分别随机选择GF(2^w)上的多项式$f_i(x)k_i + a_1^{(i)}x + a_2^{(i)}x^2 + \cdots + a_{t-1}^{(i)}x^{t-1}$，$g_i(x) = c_i + b_1^{(i)}x + b_2^{(i)}x^2 + \cdots + b_{t-1}^{(i)}x^{t-1}, i = 1,2,\cdots$。$D$通过秘密信道把$f_i(\alpha_j)$，$g_i(\alpha_j)$发送给$P_j$作为$P_j$持有的关于密钥字$k_i$和密文字$c_i$的份额，$j$=1,2,…,$n$，$i$=1,2,…。最终，分享者$P_j$得到的关于密钥流$K$和密文流$C$的份额序列分别为$f_1(\alpha_j)\ f_2(\alpha_j)\ f_3(\alpha_j)\ \cdots$和$g_1(\alpha_j)\ g_2\alpha_j)\ g_3(\alpha_j)\ \cdots$。

计算明文流的份额序列 由Shamir秘密分享方案的线性性可知，分享者P_j只须将自己所持有的关于密钥流K的份额序列与关于密文流C的份额序列对应相加即可得到关于明文流M的份额序列$h_1(\alpha_j)\ h_2(\alpha_j)\ h_3(\alpha_j)\ \cdots$，其中$h_i(\alpha_j) = f_i(\alpha_j) + g_i(\alpha_j)$，$i$=1,2,3,…，$j$=1,2,…,$n$。 (这里的加法是GF($2^w$)上的加法运算)。

分布式的解密 n个分享者P_1，P_2 …，P_n中的t个或t个以上成员利用他们持有的关于明文流的份额序列，按照Shamir (t,n)门限方案的恢复算法可恢复出明文序列$M= m_1m_2m_3\cdots$。

以t个成员P_1，P_2 ,…,P_t合作解密为例，我们给出计算明文序列的公式[1, 6]：

$$m_k = \sum_{i=1}^{t} h_k(\alpha_i) \prod_{\substack{j=1 \\ j\neq i}}^{t} \frac{-\alpha_j}{\alpha_i - \alpha_j}, \qquad k = 1,2,3,\cdots \tag{1}$$

上式中的所有运算都在GF(2^w)中进行。

这样便实现了分布式解密。需要说明的是，在分享者进行合作解密之前，每一个分享者无法得知密钥流序列、密文流序列和明文流序列的任何一个字。

5 应用实例

下面结合基于字的流密码的分布式解密给出引言中例子的一个具体解决方案。

(1) 背景 将该公司网络用 1 个数据库，6 个服务器和一些终端组成。每个终端均与所有的服务器相连，如图 3 所示。在这个体制中还需有一个密钥流生成器。所有公司雇员都被授权访问数据库。这并行的 6 个服务器是用来进行分布式解

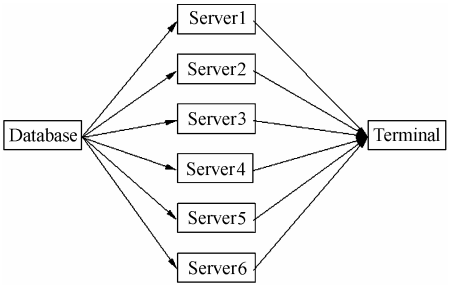

图 3　分布式解密实例

Fig.3　An instance of distributed decryption

密的。明文已事先被加密成密文存放在数据库里(选用一种基于字的流密码体制来加密，如 RC4)。

设明文$M=m_1m_2\cdots=$“10”“20”,…(十六进制表示)，基于字(简单起见，不妨设字长为 8)的密钥流生成器产生的密钥$K=k_1k_2\cdots=$“11”“14”…，则相应的密文为$C=c_1c_2\cdots=$“01”“34”…。

(2) 密钥字序列与密文字序列的秘密分享　由上述背景知，可选择GF(2^8)上的(3,6)门限方案。首先确定GF(2)上的一个 8 次不可约多项式为$m(x)=x^8+x^4+x^3+x+1$。关于有限域上不可约多项式的判定算法可参见文献[7]。数据库的密文经(3,6)门限方案SS变换后产生的密文份额序列分发给每一个服务器，同样地，密钥流生成器产生的密钥流经(3,6)门限方案SS变换后产生的密钥份额序列也分发给每一个服务器(如图 4 所示)。

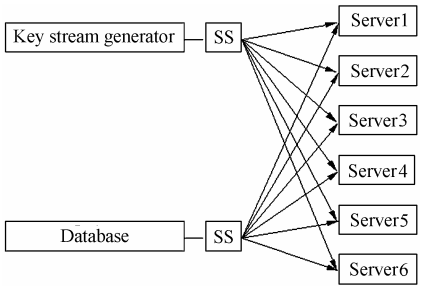

图 4　密钥与密文的分享

Fig.4　The sharing of keys and ciphers

对于密钥字序列的每一个字k_i，创建一个GF(2^8)上的二次多项式：$f_i(x)= k_i+a_ix+b_ix^2$ 。其中a_i，b_i是从GF(2^8)中随机选择的。利用GF(2^8)上的加法与乘法运算计算出f_i(“0j”)，把它作为k_i的第j个份额传送给服务器j(j=1,2,…,6)。现假定已随机选择的系数如下：$A=(a_i)=$“1B”“03” … ；$B=(b_i)=$“50”“61”…。则 k_1的秘密分享多项式就是$f_1(x)=$ “11” +“1B”x+“50”x^2，于是传送给 6 个服务器的密钥字k_1的份额依次是f_1(“01”)= “5A’，f_1(“02”)= “7C ”，f_1(“03”)= “37”，f_1(“04”)= “0A”，f_1(“05”)= “41”，　f_1(“06”)= “67”。类似可得：f_2(“01”)= “76”，f_2(“02”)= “8D”，f_2(“03”)= “EF”，f_2(“04”)= “52”，f_2(“05”)= “30”，f_2(“06”)= “c7”。服务器 1,2, …,6 所收到的密钥K的份额序列依次是：

$$F_1=(f_i(\text{“01”}))=\text{“5}A\text{” “76” }\cdots$$

$$F_2=(f_i(\text{“02”}))=\text{“7}C\text{ ” “8}D\text{” }\cdots$$

$$F_3=(f_i(\text{“03”}))=\text{“37” “}EF\text{” }\cdots$$

$$F_4=(f_i(\text{“04”}))=\text{“0}A\text{” “52” }\cdots$$

$$F_5=(f_i(\text{“05”}))=\text{“41” “30” }\cdots$$

$$F_6=(f_i(\text{“06”}))=\text{“67” “c7” }\cdots$$

对于密文字序列的每一个字c_i，也类似创建一个GF(2^8)上的二次多项式：$g_i(x)= c_i+u_ix+v_ix^2$ 。 假定已随机选择的系数如下：$U=(u_i)=$ “2A” “13” … ；$V=(v_i)=$ “21” “45” … 则服务器 1,2,…,6 所收到的密文C的份额序列依次是：

$$G_1=(g_i(\text{“01”}))=\text{“0}A\text{” “62” }\cdots$$

$$G_2=(g_i(\text{“02”}))=\text{“}D\text{1” “1}D\text{” }\cdots$$

$$G_3=(g_i(\text{“03”}))=\text{“}DA\text{” “4}B\text{” }\cdots$$

$$G_4=(g_i(\text{“04”}))=\text{“8}F\text{” “44” }\cdots$$

$$G_5=(g_i(\text{“05”}))=\text{“84” “12” }\cdots$$

$$G_6=(g_i(\text{“06”}))=\text{“5}E\text{” “6}D\text{” }\cdots$$

(3) 解密　现在，每个服务器都收到两个份额序列，一个来自密钥流生成器，而另一个来自数据库。基于这两个序列，服务器可“解密”密文，也就是说，它只需将收到的两个序列对应相加(在GF(2^8)上做加法运算)即得明文M的份额序列。易得服务器 1,2, …,6 所计算出的明文M的份额序列依次是：

$$H_1=(h_i(\text{“01”}))=\text{“50” “14” }\cdots$$

$$H_2=(h_i(\text{“02”}))=\text{“}AD\text{” “90” }\cdots$$

$$H_3=(h_i(\text{“03”}))=\text{“}ED\text{” “}A4\text{” }\cdots$$

$$H_4=(h_i(\text{“04”}))=\text{“85” “16” }\cdots$$

$$H_5=(h_i(\text{“05”}))=\text{“}C5\text{” “22” }\cdots$$

$$H_6=(h_i(\text{“06”}))=\text{“39” “}AA\text{” }\cdots$$

当有用户通过终端向各服务器发送请求访问数据库时，各服务器均首先对请求作检查(如采用身份认证等一系列访问控制技术)，确认是合法请求时，再分别向数据库、密钥流生成器调取密文、密钥，最后，各服务器均会将自己“解密”的明文M的份额序列传送给终端。这样，在终端便可通过拉格朗日插值法恢复出明文。例如，要恢复出明文字m_1，在终端只需明文字m_1的 6 个份额中的任 3 个即可利用公式(1)计算出明文m_1=“10”。用同样方法可得m_2=“20”，于是在终端便恢复了明文$M= m_1\ m_2\cdots=$“10” “20” …。

6　安全性与效率

本文所述的基于字的流密码分布式解密方案(以下简称新方案)的安全性主要基于两方面: 一是所选用的加密算法的安全强度；二是所选择的秘密共享体制的安全性。与传统方案相比，新方案的安全性有了显著提高。因为，在传统方案中，解密是由单个机器完成的，无论是由哪个机器来完成

解密，该机器必将成为攻击者的众矢之的，是整个系统的脆弱点。而新方案利用门限密码技术将解密密钥及密文在一组机器间进行秘密分享，并且将解密操作分散由一组机器来共同合作完成，使参于解密的任一机器都不知道解密密钥、密文及明文，避免了脆弱点的存在。此外，在新方案中，攻击者至少要同时在t 个通信信道上窃听成功才有可能重建密文(在数据库与服务器之间的信道上窃听)或明文(在服务器与终端之间窃听)，而攻击者要使拒绝服务(DOS)攻击凑效，则他必须能控制$n-t$个服务器，使之不能正常工作。这些无疑都增加了攻击者实施攻击的难度。与简单加法流密码的分布式解密方案(以下简称原方案)相比，新方案提供了更高的安全保障。这是因为原方案所选用的加密算法是简单加法流密码，而新方案选用的加密算法是基于字的流密码(如：RC4，SNOW等，其安全性分析可参见文献[3,4])，就加密算法本身的安全性而言，新方案的安全强度已有了明显提高。另外，原方案所选用的秘密共享体制是GF(2)上的Shamir秘密分享方案，而新方案选用的秘密共享体制是GF(2^w)上的Shamir秘密分享方案，就秘密分享方案的安全性而言，新方案的安全性也有一定的提高。关于原方案的安全性，Magnus Öberg在文献[2]中已作了详细的分析，此处不再重述。注意到在分布式解密方案中，秘密共享体制本身的安全性是整个方案安全性的重要方面，因此，在新方案中，还可通过选择可证实的秘密共享方案[8]来进行密文流及密钥流的秘密分享以进一步提高整个方案的安全性。

效率方面，原方案的加密操作及秘密分享都是按位进行的，即一次只能处理 1bit字符，效率很低。新方案的加解密操作及秘密分享都是在同一个有限域(即GF(2^w))上进行的，所有的运算都是按字进行的，即一次处理 1 个字(字长为w的比特串，w可取 8，16，32 等等)，相比之下，效率已有了很大的提高。新方案的实现效率取决于两方面，一是加密算法的实现效率，这是由所选用的加密算法决定的。目前，关于基于字的流密码的实现已有许多结果，例如文献[4]就分析了流密码SNOW的软、硬件实现效率。一般而言，流密码的实现效率及加解密速度要远比分组密码高，与公钥密码相比，那更是“不可同日而语”了。流密码正是以易于实现、加解密速度快而著称。二是秘密分享方案的实现效率。新方案采用的是GF(2^w)的Shamir秘密分享方案，该方案分配算法的复杂度是$O(n)$，恢复算法的复杂度是$O(n^3)$，所需要的运算是有限域上的加法和乘法运算(其软件实现可参见文献[7])。Shamir秘密分享方案是迄今为止最易于实现而安全性又较高的一种秘密共享体制。可证实的秘密共享虽然安全性很高，但效率较低。由上分析可知，就目前而言，新方案的实现效率是较为理想的。

7 结束语

上述实例表明对于基于字的流密码实现分布式解密是可行的。在本文所述方案中，加解密操作都是在GF(2^w)上进行的。显然，加解密运算所依赖的域的阶不再要求是素数。新方案较原方案相比，具有安全、高效和广泛适用性的特点。可用于对目前公认的安全高效的一些流密码算法诸如RC4，SNOW等进行分布式解密，也可直接应用于诸如数据库信息的保护，还可应用于Ad-hoc网中分布式密钥管理[9]。

注意到在新方案中，需要向每一个服务器传送密钥的份额序列，如果能像文献[2]中所提出的改进方案那样，只将种子密钥的份额序列传送给每个服务器，而其余的密钥份额序列可由各个服务器独立产生，势必会减少一定的通信量。遗憾的是，基于字的流密码的密钥流生成器一般都是由驱动部分和非线性性变换部分组成，要想实现文献[2]中的改进方案，则非线性性变换部分的多项式表示是必需的。这方面的工作还有待于进一步研究。

参 考 文 献

[1] Shamir A. How to share a secret. *Communications of the ACM*, 1979, 24(11): 612 – 613.

[2] Magnus Öberg. Distributed stream ciphers. Link Öping Studies in Science and Technology Thesis No. 1021.

[3] Rick Wash. Lecture Notes on Stream Ciphers and RC4. Available at: http://www.crimelabs.net/docs/stream.pdf. 2004: 5 – 12.

[4] Ekdahl P, Johansson T. SNOW-a new stream cipher. NESSIE project submission, 2000.

[5] 胡予濮, 张玉清, 肖国镇. 对称密码学[M]. 北京: 机械工业出版社, 2002: 218 – 220.

[6] 王育民, 刘建伟. 通信网的安全----理论与技术[M]. 西安: 西安电子科技大学出版社, 1999: 102 – 182.

[7] 周玉洁, 冯登国. 公开密钥密码算法及其快速实现[M]. 北京: 国防工业出版社, 2002: 35 – 87.

[8] Gennaro R. Theory and practice of verifiable secret sharing. [Ph.D. thesis], Massachusetts Institute of Technology, 1996.

[9] Kong J, Zerfos P, Luo H, *et al*.. Providing robust and ubiquitous security support for mobile Ad hoc networks. Availableat: http: //www.cs.ucla.edu/wing/pdfdocs/ICNP01.pdf. 2004: 5 – 18.

刘志高: 男，1975 年生，讲师，研究方向为密码学、网络安全.
张福泰: 男，1965 年生，教授，研究方向为信息安全及电子商务.
徐 倩: 女，1980 年生，硕士生，研究方向为密码学、网络安全.