对一种 RSA 改进算法的安全性分析①

夏 伟[1,2], 潘 瑜[1]

[1](江苏理工学院 计算机工程学院, 常州 213001)
[2](青海师范大学 计算机学院, 西宁 810000)

摘 要: RSA 和背包公钥密码算法都是经典的加密算法, 目前仍有很多密码研究者在研究它们的改进算法. 王茜等作者将两者结合起来设计了一个新的密码方案, 将 RSA 用到了背包密码体制中. 对这一新方案进行了安全性分析, 从三个角度对这一方案进行了分析说明, 并通过格规约攻击计算实验来验证, 最终说明这种改进之后的公钥密码算法仍然是不安全的.

关键词: 背包公钥密码; 改进算法; 安全性分析; 格规约

Security Analysis of an Improvement of RSA Public-Key Cipher Scheme

XIA Wei[1,2], PAN Yu[1]

[1](School of Computer Engineering, Jiangsu University of Technology, Changzhou 213001, China)
[2](School of Computer Science, Qinghai Normal University, Xining 810000, China)

Abstract: RSA and Knapsack public-key cipher algorithm are classical encryption algorithms. There are also many people researches about the improvement of them recently. Wang Qian created a knapsack public-key cipher scheme based on RSA. There are a security analysis of the improved algorithm and a analysis form three angles of the scheme in this paper. Finally, we will use the calculation results of the lattice code against to prove that the improved algorithm is still not safe.

Key words: knapsack public-key cipher; improved algorithm; security analysis; lattice reduction

自 1976 年美国密码学专家 Diffie 和 Hellman[2]提出公钥密码体制以来, 公钥密码体制一直是密码学研究的热点, 而公钥密码体制中又以 RSA[3]和背包公钥密码[4]最为人们所熟知. 虽然这两种算法以及它们的一些改进算法[5-8]已被破解[9-11], 但随后仍有不少其他改进方案被提出来. 既然基于大整数分解和基于背包的公钥密码体制不安全, 为什么仍有大量的研究人员在研究它们的改进方案呢? 根据研究发现: 首先, 背包密码体制加解密非常迅速, 非常适合现实中的应用; 其次, RSA 是基于大整数分解的, 背包密码本身是基于背包问题的, 它们都具有 NPC(NP 完全性)特性, 很适合应用到加密体制中; 最后, 背包公钥算法非常适合现实中有很多资源受限制的应用环境, 比如内存受限, 时间受限等等. 所以, 对 RSA 以及背包密码体制的研究以及改进方案依然源源不断, 文献[1]就是其中一个.

文献[1]对 RSA 以及 M-H 背包算法进行了研究和安全性分析, 并在此基础上将背包公钥密码的思想与 RSA 算法结合在一起, 形成了一种新的加密方案, 为公钥密码的研究提供了新的思路. 但这种加密方案的安全性还未被研究与证明.

通过对文献[1]中提出的一种新的加密方案的安全性进行分析, 并对其进行了高密度的格规约攻击计算实验, 从而发现这个新的加密方案仍然是不安全的.

1 基于RSA算法的背包密码体制

1.1 算法介绍

文献[1]中分别对 RSA 以及 M-H 背包算法进行了

① 基金项目:国家自然科学基金(61302124)
收稿时间:2013-11-13;收到修改稿时间:2013-12-23

安全性分析, 说明了 RSA 不能抵抗中间人攻击, M-H 背包算法也已经被 Shmir 等人攻破了. 他们提出了将 RSA 算法与背包密码思想相结合的方法, 从而设计出一个新的加密算法.

1.2 算法描述

1.2.1 密钥生成算法

1)随机选取 2 个不同的大素数 p 和 q, 两数长度大致相等;

2)计算 $n=pq$;

$$\varphi=(p-1)(q-1); \tag{1}$$

3) 随机选取个整数

$$e_1,e_2,\cdots,e_m,m<<n,$$

$$1<e_i<\varphi,1\le i\le m \tag{2}$$

$$e_i\ne e_j,1\le i\ne j\le m,$$

$$\gcd(e,\varphi)=1; \tag{3}$$

4)用欧几里得扩展算法计算 $d,1<d_i<\varphi$, 满足

$$e_id_i=1(\mathrm{mod}\,\varphi),1\le i\le m; \tag{4}$$

5) 选取 m 组非超递增不等背包序列 $(b_{k1},b_{k2},\cdots,b_{kn}),1\le k\le m,$ 满足

$$\sum_{i=1}^{n}b_{ki}x_{ki}=e_k,1\le k\le m; \tag{5}$$

6)公钥为 n;

$$(x_{k1},x_{k2},\cdots,x_{kn})\quad(b_{k1},b_{k2},\cdots,b_{kn}),1\le k\le m,$$

作为身份验证码;

私钥为 d_i, $1\le i\le m$. (6)

1.2.2 加密算法

为将明文 m 变换为密文 c, 执行以下步骤:

1)计算 $\sum_{i=1}^{n}b_{ki}x_{ki}=e_k$, (7)

将 m 分解成长度小于 $\log_2 n$ 的数据分组;

2)计算 $c=m^{e_k}\ \mathrm{mod}\,n$; (8)

1.2.3 解密算法

为将密文 c 恢复成明文 m, 执行以下步骤:

1)计算 $\sum_{i=1}^{n}b_{ki}x_{ki}=e_k^{'}$

和 $d_k^{'}e_k^{'}=1\mathrm{mod}\,\phi$, (9)

求得 $d_k^{'}$, 判断 $d_k^{'}$ 是否等于 d_k, 如果相等, 则继续下一步, 如果不等, 则放弃执行;

2)计算 $m=c^{d_k}\ \mathrm{mod}\,n$ (10)

得到明文 m.

2 安全性分析

在上述新算法中, 由于 e 是在加密方通过 $(x_{k1},x_{k2},\cdots,x_{kn})$ 和 $(b_{k1},b_{k2},\cdots,b_{kn})$ 计算得到的, 并没有像 RSA 中将 e 作为公钥公开, 从而攻击者只能获得 n, c 和 $(x_{k1},x_{k2},\cdots,x_{kn})$. 文献[1]指出攻击者要破解此算法的三个难点: 第一, 不仅要分解 n, 还要通过 $(x_{k1},x_{k2},\cdots,x_{kn})$ 求得 e_k, 而根据 $\sum_{i=1}^{n}b_{ki}x_{ki}=e_k$ 可知, 攻击者已知 $(x_{k1},x_{k2},\cdots,x_{kn})$ 和 $(b_{k1},b_{k2},\cdots,b_{kn})$ 两者中任何一个都无法在多项式时间内求解出 e_k; 第二, 在 e_k 生成 d_k 后便被消除, 解密方保存有 d_k, 而通过 d_k 求解 e_k 必须解同余式 $e_id_i=1(\mathrm{mod}\,\varphi)$, 因此攻击者想要得到 e_k 又必须获得 φ, 这也是很困难的; 第三, 攻击者如果获得 e_k 和 $(x_{k1},x_{k2},\cdots,x_{kn})$, 求解 $(b_{k1},b_{k2},\cdots,b_{kn})$ 即为求解一个非超递增不等背包序列问题, 这是个 NPC 问题.

本文针对以上提出的第一个难点, 考虑到在实际应用中 n 的长度不能太长, 及不能超过 1024, 不然运算量太大, 不利于实际应用, 而传统 RSA 已被攻破, 在 n 的长度小于 1024 时可以分解 n, 得到 p 和 q, 从而再根据公式(2)即可得到 φ; 针对第二个难点, 文献[1]指出攻击者要得到 e_k 又必须获得 φ, 而 φ 在上述过程中是可以得到的, 所以攻击者也可以获得 e_k; 针对第三个难点, 文献[1]提出, 求解非超递增不等背包序列问题是个 NPC 问题, 但笔者认为此类问题是限定条件下的线性不定方程问题, 通过构造合适的格, 只根据 $(x_{k1},x_{k2},\cdots,x_{kn})$ 和 e_k, 是可以求解或恢复部分 $(b_{k1},b_{k2},\cdots,b_{kn})$ 的.

下面我们主要分析针对第三个难点的格规约攻击:

由于 $\sum_{i=1}^{n}b_{ki}x_{ki}=e_k$, e_k 是 $\sum a_ix_i$ 的形式, x_i 都是 0 或 1, 所以 x_i 与 e_k 之间的关系总可以模拟出来, 而 LLL 算法正可以进行这种模拟, 例如构造一个格:

$$L=\begin{pmatrix}-e_k & -1 & -1 & \cdots & -1 & 1\\ b_{k1} & 2 & 0 & \cdots & 0 & 0\\ b_{k2} & 0 & 2 & \cdots & 0 & 0\\ \vdots & \vdots & \vdots & & \vdots & \vdots\\ b_{kn} & 0 & 0 & \cdots & 2 & 0\end{pmatrix} \tag{11}$$

由上面构造的格 L 的形式, 进行 LLL 规约后, 如果存在首项为 0, 尾项为 1 的规约基, 必定有 $b_{k1}x_{k1}+b_{k2}x_{k2}+\cdots+b_{kn}x_{kn}=e_k$ (这里的 $(b_{k1},b_{k2},\cdots,b_{kn})$ 是规约过程中的系数). 若中间项为 1 时, 对应行的系数为 1; 若中间项为-1 时, 对应的行的系数为 0, 则此时 $(b_{k1},b_{k2},\cdots,b_{kn})$ 就是我们要得到的.

为了进一步说明文献[1]中的新算法仍然存在安全威胁, 下面根据文献[12]中的方法对这一新算法的第三个难点进行攻击实验, 以实验数据来证明这一点.

3 格规约攻击方法

3.1 格规约攻击的算法设计

1)根据算法描述可以求出 e_k, x_{ki}, 现在我们构造格 L 如下:

$$L=\begin{pmatrix} -e_k^{\,2} & -1 & -1 & \cdots & -1 & 1 \\ b_{k1}*e_k & 2 & 0 & \cdots & 0 & 0 \\ b_{k2}*e_k & 0 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ b_{kn}*e_k & 0 & 0 & \cdots & 2 & 0 \end{pmatrix} \tag{12}$$

2)调用 LLL 算法, 得到格 *L* 的规约基 *A*.

3)在规约基 *B* 中寻找首项为 0, 尾项为 1, 中间项为 1 或-1 的规约基, 数学表达式如公式(13)所示:

$$A_{i,0}=0,\ A_{i,n+1}=1,\ A_{i,j}=1 or -1 (i\neq j, i,j=0,1,\cdots,n+1) \tag{13}$$

4) 如果在规约基 *A* 中找到公式(13)中的向量, 则输出 $B=(b_{k1},b_{k2},\cdots,b_{kn})$, 数学表达如公式(14):

$$\begin{cases} b_{ki}=1 & (A_{i,j}=1) \\ b_{ki}=0 & (A_{i,j}=-1) \end{cases} (i,j=1,2,\ldots,n) \tag{14}$$

分析此启发式格规约攻击所需要花费的时间:

调用 LLL 算法需要的时间为 $n^6\log^3(n\|A\|_\infty)\approx n^6\log^3 n$, 实际的运行时间远远小于这个时间, 而扫描找到需要的那个向量所要花费的时间为 $O(n^2)$, 所以此启发式格规约攻击所需要花费的时间在多项式时间内, $n=1024$ 的时候大约 7 个多小时就可以得到结果.

3.2 格规约攻击计算实验

为了进一步说明上述启发式格规约攻击的有效性, 可以利用 NTL[13]算法库来进行计算实验验证, 限于文章篇幅, 这里仅通过 n=16 来进行验证说明.

具体参数值如下:

非超递增序列 b_{k1}=[2 7 3 28 15 57 113 227 1813 453 907 3627 7253 29018 58037 14510];

e_1=76286;

x_{k1}=[1 1 1 1 1 1 0 0 0 0 0 1 0 0 1 1];

背包密度 density = $n/|m|$=0.9411764706

测试结果如下:

LLL(A)=

[[0 4 -2 2 0 0 0 0 0 0 0 0 0 0 0 0]
[0 2 -4 -2 0 2 0 0 0 0 0 0 0 0 0 0]
[0 2 0 0 2 -4 0 0 0 0 0 0 0 0 0 0]
[0 -2 -4 2 2 -2 0 0 0 0 0 0 0 0 0 0]
[0 0 -4 0 -2 -2 2 0 0 0 0 0 0 0 0 0]
[0 0 0 0 -4 0 -2 2 0 0 0 0 0 0 0 0]
[0 1 1 1 1 1 1 -1 -1 -1 -1 -1 1 -1 -1 1 1 1]
[0 2 0 -2 0 0 0 -4 2 0 0 0 0 0 0 0]
[0 -2 0 2 0 0 0 0 -4 0 2 0 0 0 0 0]
[0 0 0 0 0 0 0 0 -4 0 -2 2 0 0 0 0]
[0 -1 -1 -1 -1 -1 -1 1 -3 -1 -1 -1 1 1 1 -1 -1 -1]
[0 3 -1 -3 -1 -1 -1 1 -3 3 -1 -1 -1 1 1 -1 -1 -1]
[0 -1 -1 -1 -1 -1 -1 1 -1 -1 -1 -1 -3 1 -1 1 1 1]
[0 0 0 2 0 0 0 0 2 0 0 0 0 2 -2 2 0 2]
[0 0 2 -2 0 0 0 0 2 0 0 0 0 2 2 0 0 2]
[0 -2 0 2 0 0 0 0 2 0 0 0 0 -2 0 2 -2 2]
[-76286 2 0 -2 0 0 0 0 0 0 0 0 0 0 0 0 0]
]

根据公式(13)我们寻找发现, 规约基中存在首项为 0, 尾项为 1, 中间项为 1 或-1 的向量:

[0 1 1 1 1 1 1 -1 -1 -1 -1 -1 1 -1 -1 1 1 1]

再根据公式(14)我们得到恢复出的序列为:

[1 1 1 1 1 1 0 0 0 0 0 1 0 0 1 1]

正好与 x_{k1} 吻合, 从而说明攻击成功.

这只是 *n* 为 16 位的情况, 而当 *n* 为很大的数时, 此攻击方法就不能够完全恢复明文, 只能恢复部分明文, 所以还没有最终攻破此算法, 但通过以上的安全性分析, 不难发现此密码方案还是存在着安全隐患, 还需要引起足够的重视.

4 结语

本文对文献[1]中新算法的安全性进行了分析，并对其中使用背包思想的部分，根据不同的参数，在高背包密度下对其进行了启发式的格规约攻击，攻击时间均在多项式时间内，通过计算实验验证了格规约攻击的有效性，从而说明这种新算法仍然存在潜在的漏洞与威胁. 虽然基于背包的密码方案有快速加解密的优点，但鉴于背包体质本身的特性，不管是使用超递增序列还是非超递增序列，都会为攻击者留下一些的冗余信息，总可以通过格规约攻击以一定的概率恢复出对攻击者有用的信息，存在不可忽略的威胁.

参考文献

1 王茜,倪建伟.一种基于 RSA 算法的加密算法.重庆大学学报(自然科学版),2005,28(1):68−72.

2 Diffie W, Hellman ME. New directions in cryptography. IEEE Trans. in Information Theory, 1976, 22(6): 644−654.

3 Rivest RL, Shamir A, Adleman LM. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 1978, 21(2): 120−126.

4 Merkle RC, Hellman MH. Hiding information and signatures in trapdoor knapsacks. IEEE Trans. on Information Theory, 1978, 24(5): 525−530.

5 王保仓,胡予濮.高密度背包型公钥密码体制的设计.电子与信息学报,2006,28(12):2390−2393.

6 张卫东,王保仓,胡予濮.一种新的背包公钥密码体制的设计.西安电子科技大学学报,2009,36(3):506−511.

7 李云飞,柳青,李彤,等.一种可有效并行的 RSA 算法的研究.计算机应用研究,2011,28(11):4345−4349.

8 石井,吴哲,谭璐,等.RSA 数据加密算法的分析与改进.济南大学学报,2013,27(3):283-286.

9 韩立东,刘明洁,毕经国.两种背包的公钥密码算法的安全性分析.电子与信息学报,2010,32(6):1485−1488.

10 Youssef AM. Cryptanalysis of a knapsack-based probabilistic encryption scheme. Information Sciences, 2009, 179(18): 3116−3121.

11 潘彦丰,杨卫武.对一种基于 Euler-Fermat 小定理的背包公钥系统的攻击.信息工程大学学报,2011,12(5):532−534.

12 古春生,于志敏,景征俊.基于随机背包公钥密码的格攻击.计算机应用研究,2012,(9):3486−3488

13 Shoup V. NTL. a library for doing number theory. http://shoup.net/ntl/. [2009-08-14].