关于 RSA 加密算法及差分错误注入攻击的原理分析

冯　旭

（重庆电子工程职业学院　重庆　401331）

［摘　要］ RSA 加密是一种广泛应用在一线工厂的加密算法，该种加密算法自诞生以来一直备受各界关注，各种攻击方式层出不穷。这些攻击方式也在一定程度上威胁到了算法的安全性，这也是 RSA 加密算法的秘钥长度一直在增加的原因。本文将简述 RSA 加密算法的基本数学原理，以及针对于 CRT-RSA 的差分错误攻击算法的原理。

［关键词］ RSA 加密算法、差分错误注入

引言

传统的加密算法都是对称加密，这种加密方式安全性不高，因此，采用了非对称加密。在实际运用中，为了提升运算速度一般都采用了使用了中国剩余定理的 CRT-RSA。本文中将要讨论的攻击算法也被成为 Bellcore 攻击算法，本文将会给出该算法的基本数学模型。

RSA 加密算法

RSA 加密算法实际上就是指数运算与取模运算的结合，只是其中对特定元素有特定的要求。对于一个 RSA 加密算法有如下几个过程：

1. 选择一对不同的、足够大的素数 p，q。

2. 计算 n=pq。

3. 计算 f(n)=(p-1)(q-1)，同时对 p, q 严加保密，不让任何人知道。

4. 找一个与 f(n) 互质的数 e，且 1<e<f(n)。

5. 计算 d，使得 de ≡ 1 mod f(n)。这个公式也可以表达为 d ≡ e-1 mod f(n)

这里要解释一下，≡是数论中表示同余的符号。公式中，≡符号的左边必须和符号右边同余，也就是两边模运算结果相同。显而易见，不管 f(n) 取什么值，符号右边 1 mod f(n) 的结果都等于 1；符号的左边 d 与 e 的乘积做模运算后的结果也必须等于 1。这就需要计算出 d 的值，让这个同余等式能够成立。

6. 公钥 KU=(e,n)，私钥 KR=(d,n)。

7. 加密时，先将明文变换成 0 至 n-1 的一个整数 M。若明文较长，可先分割成适当的组，然后再进行交换。设密文为 C，则加密过程为：$C \equiv M^e \pmod{n}$。

8. 解密过程为：$M \equiv C^d \pmod{n}$

根据上述算法，和非对称加密算法的特性，公钥是公开的，也就是说一个第三方能知道 e 和 n 的值。攻击者的目标是 d 的值，而 d 的值又由 f(n) 决定，而 f(n) 又由 p 和 q 决定，而 n 是 p 和 q 的乘积。前面也说了，n 的值是很容易获取的，p 和 q 作为 n 的因式分子最大的特色是都是素数。因此到最后破译一个 RSA 加密算法的难度就是对 n 做因式分解的难度。

而当 p 和 q 是一个大素数的时候，从它们的积 pq 去分解因子 p 和 q，这是一个公认的数学难题。虽然随着物理硬件的发展，计算机的处理速度也在飞速的增长，RSA 的秘钥长度也跟着在增长。但是相对于传统的加密算法，RSA 加密算法下秘钥长度的小幅增长就能够提供呈指数增长的安全性能。这就是为什么 RSA 加密自问世以来，在虽然一直备受黑客关注，各种攻击算法层出不穷的情况下依然能够被大众接受和广泛使用的原因。

差分错误注入攻击

在这里，由于中国剩余定理（CRT）的广泛应用，我们将讨论针对于 CRT-RSA 算法的差分错误注入攻击。

攻击原理

攻击算法的基本原理是在加密的某个指数运算过程中产生了一个随机的错误，致使最后得到一个错误的加密结果。通过这个错误的结果，再加上正确的结果，以及很容易获得的模数 N 就能够破解 RSA 加密算法。

根据 Garner 给出的计算密文的公式：$c = S_q + ((S_p - S_q)*q_{inv} \bmod p)*q$　(1)

对于一个大于 1 的整数，一定可以分解为素数的乘积，如果将相同的素数合并，得到一个关于 a 的标准分解式为：$a = p_1^{a1} p_2^{a2} \cdots\cdots p_k^{ak}$，$p_1<p_2<\cdots\cdots<p_k$（2）

其中 pi 是素数，$a_i >= 1$，i = 1,2,3……,k。

如果某个数 T 是 a 的因子，则 T 可以分解为：$T = p_1^{\beta 1} p_2^{\beta 2} \cdots\cdots p_k^{\beta k}$，$p_1<p_2<\cdots\cdots<p_k$　(3)

其中 pi 是素数，$0 \le \beta_i \le ai$，i = 1,2,3……,k

根据上面说到的错误注入，以及给出的相关公式，在运算到 $(Sp-Sq)*q_{inv} \bmod p$ 后需要接着进行下面的乘法和加法运算，令 $T = (Sp - Sq)*q_{inv} \bmod p$，因此（1）式变为：

$c = Sq + T*q$　　(4)

然后假设在加载 q 的过程中，能使 q 发生错误，同时假设两次运算是对相同的数据进行加密，在 T 和 Sq 保持不变的情况下有：

$C1 = Sq + T * q_1$　（5）

$C2 = Sq + T * q_2$　（6）

假设两次对 q_1 和 q_2 的错误注入不一样，则有：

$GCD(c-c1,c-c2) = GCD(T*q - T*q1, T*q - T*q_2) = T * GCD(q-q_1, q-q_2)$　（7）

根据上面的（4）式可知：

$q = [(c - Sq)/T] = [c/T - Sq/T]$　　（8）

[] 表示取整运算，在上式中 Sq 未知，但是如果在（7）式中得到 T 的值，根据 Sq 和 T 的关系，就能很快得到 q 的值。因此关键是获得 T 的值，根据整数唯一分解定理有：

$GCD(c-c1,c-c2) = t_1^{a1},t_2^{a2}\cdots\cdots,t_k^{ak}$

其中 t1,t2……tk 是素数，ai ≥ 1, i = 1,2,……,k, 则 T 可以表示为：

$T = t_1^{\beta 1},t_2^{\beta 2}\cdots\cdots,t_k^{\beta k}$

其中 t1,t2……tk 是素数，$\beta i \ge 0$，$\beta i \le ai$, i = 1,2,……,k

由于无法获取 Sq 的值，在计算（8）式的时候采用下列方式：

q~ = [c/T]

得到 q~ 后在其值附近进行搜索，并尝试对 N 进行分解。然后根据（8）式又可以得到：

$q\sim \ge q > Sq$

则有：$q = [c/T - Sq/T] \ge [c/T - [c/T]/T]$

令 q`=[c/T–[c/T]/T], 在区间 [q`,q~] 查找满足条件的素数即可，搜索长度为：$q`-q\sim \le [c/T^2]$, 可知，T 的值越大越容易精确定位。

总结

根据本文中对 RSA 算法的讨论可见，算法本身的数学基础并不复杂，而这个算法的安全性是体现在对一个大数做因式分解的难度。而错误注入攻击这种方式确实对算法本身造成了巨大的威胁。由上文可见，围绕着 RSA 加密和错误注入攻击的讨论是非对称加密算法领域里令人瞩目的一部分。

参考文献

[1] Aumüller C, Bier P, Fischer W, et al. Fault attacks on RSA with CRT: Concrete results and practical countermeasures[C]//International Workshop on Cryptographic Hardware and Embedded Systems. Springer, Berlin, Heidelberg, 2002: 260-275.