RSA暗号システムとCNS攻撃法

1371041F 前浜 諭

2005年2月10日

前半は公開鍵暗号系の代表であるRSA暗号系を説明し、後半はRSA暗号の攻撃法の1つであるCNS攻撃法を紹介する。この卒業論文を書くにあたり、指導していただいた松本眞先生に厚く御礼申し上げます。

1 暗号の種類

暗号には、暗号化と復号化で同じ鍵を使う共通鍵暗号方式と、暗号化するときの鍵は公開しておく公開鍵暗号方式の2種類が知られている.

共通鍵方式の利点は、暗号化及び復号化の計算時間が早いことにある. しかしながら、暗号化に用いた鍵が分かると簡単に暗号文を解読することができてしまう. 共通鍵方式では、まずやりとりする者同士で安全に鍵を交換する必要があった.

共通鍵方式での問題を解決するには、暗号化と復号化で違う鍵を用いることと、暗号化の鍵が復号化の鍵から簡単には分からないことが必要である.

暗号化の鍵から復号化の鍵が分からないならば、暗号化の鍵は公開しても暗号文を解読するのは不可能である. これが公開鍵方式の考え方である.

公開鍵方式の利点は、暗号化する鍵の管理をする必要がないことと、公開された鍵を使って誰でも暗号文を作ることができることの2点が挙げられる.

欠点は、共通鍵方式に比べて計算に時間がかかることである. そこで、現在ではメッセージを暗号化するための鍵を公開鍵方式で暗号化し、メッセージ自体は共通鍵方式で暗号化する方法が多い.

2 RSA暗号系

RSA暗号系は、1977年に、Rivest, Shamir, Adlemanによって実現した最初の公開鍵暗号方式であり、現在も広く使われている方式のひとつである.

2.1　鍵の生成

暗号化の準備として暗号化鍵および復号化鍵を作成する.

m を与えられた自然数とする。この m を暗号化することを考える. 自然数 n を 2 つの素数 $p,\ q$ を用いて

$$n = pq \qquad m < n$$

と表されるものとする.

ϕ をオイラー関数とし、$\phi(n)$ と互いに素な 1 より大きい整数を選んでそれを e とおく. $\phi(n)$ は

$$\phi(n) = \phi(pq) = \phi(p)\phi(q) = (p-1)(q-1) \tag{1}$$

と計算することができる.

d を未知数とする合同式

$$ed \equiv 1 \mod \phi(n) \tag{2}$$

について考える. e と $\phi(n)$ は互いに素であるから、この合同式は $\phi(n)$ を法としてただ 1 つの解 d をもつ. この d が復号化鍵である.

暗号のやりとりをする人は、(n, e) を公開し、(p, q, d) は秘密にしておく.

例　$p = 11,\ q = 23$ とすると、$n = 11 * 23 = 253$ となる.
$\phi(n) = (11-1)(23-1) = 220$ となるので、$e = 3$ とする.
d は、$3d \equiv 1 \mod 220$ を計算して導く.$d = 147$ となる.
$(253, 3)$ は公開し、$(11, 23, 147)$ は秘密にしておく.

2.2　暗号化

送信者は、受信者の公開された鍵 (n, e) を使って、平文 m を

$$c \equiv m^e \mod n$$

のように計算し、c が暗号文として受信者に送られる.

例　2.1 の数を用いる.
$m = 38$ を暗号化する.

$$c \equiv 38^3 \mod 253$$

を満たす c が暗号文である. $c = 224$ である.

2.3 復号化

2.3.1 復号化

受信者は送られてきた暗号文 c を、秘密にした鍵 d を用いて

$$c^d \equiv m \mod n$$

のように計算し、平文 m を得る.

例 2.2 の数を用いる.
$c = 224$ を復号化する.

$$224^{147} \equiv m \mod 253$$

を満たす m が復号化された平文である.$m = 38$ である.

2.3.2 考察

2.3.1 は次の定理を用いている.

定理 1 (n, e) を公開された鍵、d を秘密にした鍵とする. すべての m $(0 \leq m < n)$ について

$$(m^e)^d \mod n = m$$

【証明】 $ed \equiv 1 \mod \phi(n)$ より、$ed \equiv 1 \mod (p-1)(q-1)$ であるから、ed は、ある自然数 k を用いて

$$ed = 1 + k(p-1)(q-1)$$

と書くことができる.

$$(m^e)^d = m^{ed} = m^{1+k(p-1)(q-1)} = m \cdot (m^{(p-1)(q-1)})^k$$

より

$$(m^e)^d \equiv m \cdot (m^{p-1})^{(q-1)k} \equiv m \mod p \qquad \cdots (*)$$

を得る. p が m と素ならば Fermat の小定理

$$a^{\phi(n)} \equiv 1 \mod n \quad (a,\ n \text{ は互いに素})$$

より $(*)$ は成り立つ. しかし、p が m の倍数であったとしても、$(*)$ の両辺は 0 になるので、成り立つ.
同様にして

$$(m^e)^d \equiv m \mod q.$$

p と q は素数なので、

$$c^d \equiv m \mod n$$

を得る. 【証明終】

2.4 RSA の安全性

RSA 暗号の安全性は、素因数分解の難しさに由来する. つまり、攻撃者は n がわかったとしても p, q を計算することは（不可能ではないが）膨大な時間がかかる.

n の桁数として 300 桁以上とった場合、(2) 式を計算する上で、p, q を知っていれば、(1) 式より $\phi(n)$ を計算するのは容易である. 一方で、p, q を知ることなく $\phi(n)$ を効率的に計算する方法は現時点では知られていない. これより RSA 暗号の安全性が保たれる.

2.5 署名

m を送りたい平文、(n, e) を公開する鍵、d を秘密にする鍵とする.

$$r \equiv m^d \mod n$$

なる自然数 r を 1 つ選ぶ. これが m に「本人」である署名をしたものである.

両辺を e 乗すると

$$r^e \equiv (m^d)^e \mod n.$$

$ed = 1 + k\phi(n)$ であるから、

$$r^e \equiv m \cdot (m^{\phi(n)})^k \mod n.$$

よって、Fermat の小定理より

$$m \equiv r^e \mod n.$$

ゆえに、m を復号化することができる.

例 $n = 253,\ e = 3,\ d = 147,\ m = 38$ とする.
署名 r を作る.r は

$$38^{147} \mod 253$$

なる自然数であるから、$r = 113$ となる.
r から m を復元する.$m \equiv 113^3 \mod 253 = 38$ となり、復元できた.

2.6 安全性と「本人」である証明

A は (n_A, e_A) を公開し、d_A を秘密に、B は (n_B, e_B) を公開し、d_B を秘密にしているとする。

送信者 A から受信者 B にメッセージ m を秘密裏に送信するには、B の公開された鍵 e_B を用いて暗号 c_B を作り送信する.B は c_B を秘密鍵 d_B を用いて復元する。d_B を持っているのは B だけなので、m の秘密は守られる.

しかし、e_B は公開されているので、これを使って第三者 C からメッセージ m' を送信できる. C は、A のふりをして B に誤った情報を送ることができる.

A が「本人」であることを証明するには次のようにする.

A は、自分の秘密鍵 d_A を用いて送信したいメッセージ m に署名を施して r を作り、それを B に送信する. B は、送信された r を A の公開された鍵 e_A を用いて m に戻す。署名が間違っていた場合には復号化するのは不可能なので、「本人」であることが証明される。

しかしながら、もしも C が何らかの方法で r を得た場合、C も e_A を用いて m を復元することができ、m の秘密が守られない。

上の 2 つを同時に満たすには次のようにする. A は、自分の秘密鍵 d_A を用いて送信したいメッセージ m に署名を施して r を作る.r は B の公開鍵 e_B を用いて c_B を作り、これを送信する.

B は、送信された c_B を B の秘密鍵 d_B を用いて r を得る. さらに A の公開鍵 e_A を用いて復元し、メッセージ m を得る。

3 CNS 攻撃法

CNS 攻撃法は、署名変換対称データが小さな素数の積となるメッセージを大量に集め、それらのメッセージを正当な署名者に送信して署名を生成してもらい、入手した署名から別のメッセージの署名を偽造するものである.

まず、1 つ定義を定める.

定義

合成数の中で、すべての素因数が p_k 以下になる数を p_k-*smooth* という.

攻撃者は、「意味のある」メッセージを自由に偽造署名して送信することを目標とする。しかし、それはまず不可能である。そこで攻撃者は、最小の素数から数えて k 番目に大きな素数 p_k を設定して、p_k-*smooth* となるメッセージ m を大量に集める.

このとき、m は

$$m = \prod_{j=1}^{k} p_j^{v_j} \quad (p_j \le p_k,\ j = 1, \ldots, k) \tag{3}$$

を満たす.

攻撃者は、(3) 式を満たす r 個のメッセージ

$$m_i = \prod_{j=1}^{k} p_j^{v_{i,j}} \quad (p_j \le p_k,\ j = 1, \ldots, k,\ 1 \le i \le r)$$

を正当な署名者に送信する. 署名者は自分の秘密鍵を利用して

$$(m_i)^d = \prod_{j=1}^{k} p_j^{(v_{i,j})d} \mod n \tag{4}$$

で表される署名 $(m_i)^d \mod n$ を返信する.

署名偽造の対象となるメッセージを m_{r+1} とすると、m_{r+1} も p_k-*smooth* となる必要があるので、(3) 式を満たす必要がある. 攻撃者は署名偽造するメッセージを自由に選択することができないが、意味のあるメッセージの署名偽造ができる可能性もある.

攻撃者は、入手した $(m_i)^d \mod n$ を利用して、メッセージ $m' = m_{r+1}$ に対する署名 $(m_{r+1})^d \mod n$ を生成（偽造）する. そのために、$(m_i)^d \mod n$ を説明変数、$(m_{j+1})^d \mod n$ を従属変数とする方程式を以下の手順で導く.

m_i を

$$m_i \longmapsto \boldsymbol{V}_i = \{v_{i,1} \mod e,\ v_{i,2} \mod e,\ \cdots,\ v_{i,k} \mod e\}$$

のように表される k 次元ベクトル $\boldsymbol{V}_i$ に対応させる.

$\boldsymbol{V}_i$ を要素とするベクトルの集合の要素は e^k 個存在する. また、基底となるベクトルは $\{\boldsymbol{p}_1, \boldsymbol{p}_2, \cdots, \boldsymbol{p}_k\}$ である.

偽造メッセージ m_{r+1} にも、k 次元ベクトル $\boldsymbol{V}_{r+1}$ が存在し、他の r 本のベクトル $\boldsymbol{V}_i$ の線形結合で表されることが知られている. すなわち、$\boldsymbol{V}_{r+1}$ は適当な定数 b_i $(i = 1, \cdots r,\ 0 \le b_i \le e-1)$ によって

$$\begin{aligned} \boldsymbol{V}_{r+1} &= \{v_{r+1,1} \mod e, v_{r+1,2} \mod e, \cdots, v_{r+1,k} \mod e\} \\ &= \sum_{i=1}^{r} b_i \boldsymbol{V}_i \mod e \end{aligned} \tag{5}$$

と表される.

(5) 式に $\boldsymbol{V}_i = \{v_{i,1} \mod e,\ v_{i,2} \mod e,\ \cdots,\ v_{i,k} \mod e\}$ を代入する.

$$\begin{aligned} & \sum_{i=1}^{r} b_i \boldsymbol{V}_i \mod e \\ = & \sum_{i=1}^{r} b_i \{v_{i,1} \mod e,\ v_{i,2} \mod e,\ \cdots,\ v_{i,k} \mod e\} \mod e \\ = & \left\{ \sum_{i=1}^{r} b_i v_{i,1} \mod e,\ \sum_{i=1}^{r} b_i v_{i,2} \mod e,\ \cdots,\ \sum_{i=1}^{r} b_i v_{i,k} \mod e \right\} \end{aligned} \tag{6}$$

(5) 式と (6) 式の右辺を比較すると、$v_{r+1,j}$ について

$$v_{r+1,j} = \sum_{i=1}^{r} b_i v_{i,j} \mod e$$

が成り立つ. 上の式はある定数 a_j を用いて

$$v_{r+1,j} = \sum_{i=1}^{r} b_i v_{i,j} - a_j e \tag{7}$$

と表すことができる.

偽造メッセージ m_{r+1} に対する署名は

$$(m_{r+1})^d = \prod_{j=1}^{k} p_j^{(v_{r+1,j})d} \mod n. \tag{8}$$

これを m_j を用いて表すことを考える.

(8) 式に (7) 式を代入する.

$$\begin{aligned} (m_{r+1})^d &= \prod_{j=1}^{k} p_j^{(\sum_{i=1}^{r} b_i v_{i,j} - a_j e)d} \mod n \\ &= \prod_{j=1}^{k} p_j^{(\sum_{i=1}^{r} b_i v_{i,j})d} \cdot \prod_{j=1}^{k} p_j^{(-a_j e d)} \mod n. \end{aligned} \tag{9}$$

(2) 式より

$$\begin{aligned} (m_{r+1})^d &= \prod_{j=1}^{k} p_j^{(\sum_{i=1}^{r} b_i v_{i,j})d} \cdot \prod_{i=1}^{k} p_j^{(-a_j)} \mod n \\ &= \prod_{j=1}^{k} p_j^{(b_1 v_{1,j} + b_2 v_{2,j} + \cdots + b_r v_{r,j})d} \cdot \prod_{i=1}^{k} p_j^{(-a_j)} \mod n \\ &= \prod_{j=1}^{k} \prod_{i=1}^{r} p_j^{(v_{i,j})d \cdot b_j} \cdot \prod_{i=1}^{k} p_j^{(-a_j)} \mod n \\ &= \prod_{j=1}^{k} \left(\prod_{i=1}^{r} p_j^{(v_{i,j})d} \right)^{b_j} \cdot \prod_{i=1}^{k} p_j^{(-a_j)} \mod n. \end{aligned} \tag{10}$$

(10) 式に (4) 式を代入すると

$$(m_{r+1})^d = \prod_{j=1}^{k} \left((m_i)^d \right)^{b_j} \cdot \prod_{i=1}^{k} p_j^{(-a_j)} \mod n. \tag{11}$$

(11) 式が、署名を偽造するための方程式である.

参考文献

sec.2 : Intoduction To Cryptography (Jahannes A. Buchmann)

sec.3 : RSA 署名に対する新しい攻撃法の提案について (宇根正志)