公開鍵暗号（RSA暗号）解説その1　具体例の確認について

2023.09.08

最近の情報通信技術の中でも、公開鍵暗号技術はその名称がミステリアスなこともあって、広く知られた技術になっています。しかし、ベースとなる仕組みについても知られているかといえばそうでもないようです。そこで2回に分けて解説を試みました。
ググってみると沢山のサイトで説明がありますが、定性的な説明で満足できなかったり、技術的な解説でも前提を十分に説明しないまま、専門家でもない素人相手に【これでわかるだろう】的な記述が多く、理解するのに苦労します。
そこで本格的でかつ分かりやすい【高校数学の美しい物語 RSA暗号の仕組み】を参考にしながら、推奨されている前提知識を補足的に加えて、公開鍵暗号の原理について解説をしていきます。

・高校数学の美しい物語　RSA暗号の仕組み
https://manabitimes.jp/math/1146

まず公開鍵を使っての暗号化と復号化は以下の様に行われます。

1. メッセージを受け取る側の準備

- 大きな素数 p, q を生成し，$n = pq$ とする
- $(p-1)(q-1)$ と互いに素な整数 k_1 を取ってくる
- $k_1 k_2 \equiv 1 \pmod{(p-1)(q-1)}$ なる k_2 を取ってくる（→補足1）
- n と k_1 を公開する（公開鍵），k_2 は公開しない（秘密鍵）

2. メッセージを送る側の暗号化方法

- 送りたいメッセージを m とする。ただし，m は n 未満の正の整数とする
- 公開鍵を用いて m^{k_1} **を n で割った余り**を計算し，これを暗号文（C とおく）とする

3. メッセージを受け取る側の復号方法

- 暗号文 C と秘密鍵 k_2 を用いて C^{k_2} **を n で割った余り**を計算すると，実はこれがもとのメッセージに一致する（→補足2）！

（補足1，補足2については「公開鍵暗号 その2」で補足します。）

これを読んで、暗号化の流れをトレースするだけでも苦労するので、この説明に追加の解説を付け加えます。
【大きな素数がなぜ必要か？】
これは復号化すると、***n*** で割り算をした時の剰余が元の平文になるので、それ以上に大きい除数***n***が必要になると言うのが理由です。つまり【割った時の余りが平文になるので、割る数が平文よりも大きくないと平分戻せない、復号できない。】と言う理屈です。
【互いに素】
この意味は二つの数が互いに共通な約数を持たない事を言います。***15***と***8***というような関係です。
【剰余計算　合同式　（割った余り計算）】
まず、基本となる剰余計算について説明すると、これは文字通り、割り算の余りを計算することで ***7÷3*** の余りは ***1*** これを合同式で ***7≡1（mod3）***と記述します。ここで除数より一つ少ない剰余の種類があることに注意です。つまり除数が ***3*** ならば剰余は***1***，***2***の２種類ということです。だから、平文が100桁の数字で表現されている場合には、除数 ***n*** は100桁以上にならないといけません。
但し、合同式の計算はなじみがないので、できる限り、普通の計算式に直して説明していきます。

【素因数分解】
２つの素数の積を元の２つの素数に分解すること。積が大きくなればなるほど難しくなることが知られています。例えば、***101×171***の結果である***17271***から元の***101***と***171***を見つけることは、計算機を使っても虱つぶしに計算をする必要があり、桁数が大きくなればなるほど時間がかかります。（注1）
（注1）ネット上のサイトでは2048ビット(=617桁）の ***n*** を使って、暗号化しています。この桁数では暗号を破るのに、スパコンでも数億年かかるそうです。
【累乗計算】
これは良く知られているように、***2***の***3***乗は***8***と言う計算のこと。乗数が大きくなるとパソコンでも大変です。
【公開鍵暗号の準備】
暗号化の手順が理解ができたところで、具体的に数字を決めて動きを確認します。結構、試行錯誤をします。なんでこんな手順を踏むのか？と疑問があるでしょうが、少々辛抱。次回「公開鍵暗号その2」でその解答が分るとすっきりします。実際に計算してみると「へぇ」と思うと同時に大変だという事を感じれば十分です。

まず、**暗号受信者＝暗号作成者**は二つの**素数**を選び**、それを *p* と *q* とします。**これは一応、自由に選べます。ここで***n=p×q*** とします。次に***(p-1)・(q-1)*** **と互いに素な数** k_1を見つけます。身近な例で*p=3,q=5*とします。そうすると*(p-1)・(q-1)=8*なので k_1の候補は例えば*3*となります。*3*と*8*は共通の約数を持っていません。これまでの準備で公開鍵を作ることができました。**暗号作成者は*n＝15*と k_1 ＝ *3* を公開鍵として公開します。**そして暗号作成者は同時に $k_1 \cdot k_2 \equiv 1$（*mod(p-1)・(q-1)*）**となる** k_2 **を見つけて**おきます。
今回の例では$3 \cdot k_2 \equiv 1$（*mod8*）。つまりk_2=*3,11,19*…が候補ですが、計算が楽なので一番小さい $K_2=3$ とします。そして、*この* $k_2=3$ **は暗号作成者**だけが知っている**秘密鍵**として公開しません。つまり暗号作成者は**素数*p*と*q*の掛算結果 *n* と**それから計算できる k_1**のみを公開**します。***p,q*は公開しない**ので暗号やぶりは難しいことになります。

【具体例1】
これまで準備した数値で実際に暗号化と復号化を行ってみます。**暗号化する数字を *12*** （*12<15*）とすると***12*の*3*乗**なので***1728*。**これを*15*で割った**剰余は *3* 。暗号化された数値は *3*。これを暗号として受信者に送り**ます。これを復号するには、受け取った***3***を**秘密鍵 *3* で3**乗して ***n＝15***で割り余りを求める必要があります。**計算すると　*3*の*3*乗　割る*15*なので　*27÷15＝1*　余り*12*。**つまり**余り**が最初の**暗号化する前の数字列*12***に一致しており、復号ができました。
ここまで簡単そうに書いてますが、実際に計算してみると、うまい k_1や k_2がそう簡単に見つかりません。エクセル程度で試そうとすると結構、大変な作業です。

もう一つ例題を作ってみます。具体例１は数字が揃っているので、偶然正しいのかも知れず、別の数字でも成り立っているかを確認します。
【具体例2】
元の平文を*57*として、*p=5*, q =13　を選びます。*n=65*　*57<65*なので一応条件は満たします。***m=57です。***
ここで$k_1 \cdot k_2 \equiv 1$（*mod(p-1)・(q-1)*）なる k_1, k_2を見つけないといけません。これが結構な難物です。
$k_1 \cdot k_2 \equiv 1$（*mod48*）つまり$k_1 \cdot k_2 = 49$　***n＝65と互いに素***でないといけませんが、今回は $k_1 = k_2 = 7$ とします。
暗号文は　*57*を7乗して*65*の余剰を求めます。エクセルを使って計算すると暗号文は ***8*** となります。
復号するには *8*を7乗して*65*の余剰を求めます。同じくセクセルを使って計算すると復号文は ***57*です。**
これでめでたくもとの平文***m＝ 57***に戻すことができました。
数字の短い例しか作っていませんが、エクセル程度ではこれが限界です。ただ、公開鍵方式を使って短い数字列を暗号化し、これを共通鍵として使えば、共通鍵暗号通信を行うこともできます。

次回「公開鍵暗号その2」では、なぜこんなうまい方法があるのか、その原理である数学的な証明を紹介します。

【公開鍵暗号を再確認する】

下のエクセルを使って確認ができます。公開鍵暗号が適切に動くことを確認してください。

***p,q,k*$_1$*,k*$_2$**は意外に自由に選択ができませんので、あらかじめいくつかの数字のセットを作っておきました。

これを利用して、確かに公開鍵で暗号化、復号化ができることを確認してください。

元の数字列であるmはあまり大きくなるとエクセルにエラーが出るので、*10*程度の数字にしておくことが無難です。

パラメータ例

	p	***q***	***n***	***k*$_1$**	***k*$_2$**
セット1	5	3	15	3	3
セット2	13	5	65	7	7
セット3	17	3	51	11	3
セット4	13	3	39	5	5
セット5	29	3	87	19	3
セット6	11	3	33	7	3
セット7	17	5	85	13	5
セット8	23	3	69	9	5

①適用するパラメータを選択し、平文にnより小さい数字②を入力すると暗号文、復号文が表示されます。

適用するパラメータ

	n	***k*$_1$**	***k*$_2$**
①	69	9	5

平文　*m*	32 ②		
平文の*k*$_1$乗	35184372088832	⇐	=POWER関数
その*n*の剰余	2	⇐	=MOD関数
暗号文	2		
暗号文の*k*$_2$乗	32	⇐	=POWER関数
その*n*の剰余	32	⇐	=MOD関数
復号文	32		

以上