

認証方式

環境情報学部 4年 石原和音
学籍番号 70050607
t00069ki@sfc.keio.ac.jp
kazuto@kz-soft.com

I. RSA 署名を用いた認証とその具体例

i. 公開鍵暗号の条件

◆ 公開鍵と秘密鍵

- 公開鍵¹ PK_B B がみんなに公開。暗号化・署名の検証に利用する。
- 秘密鍵² SK_B B さんしか知らない。復号化・署名に利用する。

◆ 暗号化関数と復号関数

- 暗号化関数 E : $C = E(PK_B, M)$
- 復号関数 D : $M = D(SK_B, C)$

◆ 公開鍵暗号として必要な性質

- PK_B から SK_B を計算できない。
- PK_B と C から M を計算できない。
- SK_B と C から M を計算できる。

1 暗号化鍵ともいう。

2 復号鍵ともいう。

ii.公開鍵暗号 RSA³

▶ RSA 暗号の3つの原材料

- ◆ オイラーの φ 関数⁴
 - $\varphi(n) = |\{x \in \mathbb{N} \mid 1 \leq x \leq n \wedge \gcd(x, n) = 1\}| \quad (n \in \mathbb{N}) \dots \textcircled{1}$
 - 特に、 $n_{pq} = p \cdot q$ (p, q は素数)のとき、 $\varphi(n_{pq}) = (p-1) \cdot (q-1)$
- ◆ 剰余系 $Z_n^* = \{x \in Z_n \mid \gcd(x, n) = 1\} \dots \textcircled{2}$
 - つまり Z_n^* とは、 n を法とした剰余系 $Z_n = \{0, 1, \dots, n-1\}$ の要素のなかで、 n と互いに素である要素 x の集合である。
 - $|Z_n^*| = \varphi(n)$ ⁵
 - ちなみに、 n_p が素数であるとき、 Z_{n_p} の 0 を除くすべての要素が n_p と互いに素であり $Z_{n_p}^*$ にふくまれ、 $Z_{n_p}^* = Z_{n_p} \setminus \{0\} = \{1, \dots, n_p - 1\}$ となる。
- ◆ オイラーの定理
 - $\forall \{n, x\} \subset \mathbb{Z} \wedge 0 < n \wedge \gcd(x, n) = 1,$
 $x^{\varphi(n)} \equiv 1 \pmod{n}$
 - ちなみに、ある n_p が素数ならば、
 - $1 \leq x \leq n_p$ である x はすべて $Z_{n_p}^*$ にふくまれる。
 - $\varphi(n_p) = n_p - 1$
 - いま、 $x \in Z_{n_p}^* \Rightarrow \{n_p, x\} \subset \mathbb{Z} \wedge 0 < n_p \wedge \gcd(x, n_p) = 1$ が言えるので、オイラーの定理より
 $x^{\varphi(n_p)} \equiv 1 \pmod{n_p}$
 $x^{n_p-1} \equiv 1 \pmod{n_p}$
よって
 $x^{n_p-1} \pmod{n_p} = 1 \quad (x \in Z_{n_p}^*)$
これが先日のフェルマーの小定理である。

3 RSA 暗号にはバリエーションがあるが、ここでは、オリジナルの RSA 暗号を扱う。

4 ファイ関数、Euler's Phi function。

5 ①と②より容易に言えることである

➤ RSA 暗号と署名

- 公開鍵
 e ($e \in Z_{n_{pq}}^*$), n_{pq} ($n_{pq} = p \cdot q \wedge p$ および q は素数)
- 秘密鍵
 e^{-1} ($e^{-1} \in Z_{n_{pq}}^*$), p , q (p および q は素数)
- 暗号化関数
 $C = M^e \bmod n_{pq}$
- 復号関数
 $M = C^{e^{-1}} \bmod n_{pq}$
- 署名
 $S = M^{e^{-1}} \bmod n_{pq}$
- 署名の検証
 $M \stackrel{?}{=} S^e \bmod n_{pq}$ (右辺が左辺に等しければ正しい署名)

iii.公開鍵暗号 RSA を用いた認証の数値例

Mathematica⁶を利用してプログラムを書き、数値計算を行った。ここでは現代の情報通信の現状にあわせて、確率的素数判定法を利用して十分に大きな偽素数を利用した。有名な確率的な素数判定法には、フェルマーテストやミラー・ラビンテスト⁷があるが、今回は Mathematica の実装⁸をほぼそのまま利用した。また、e の逆元については、拡張ユークリッド互除法を用いて計算した。その結果が以下である。

(* p, q, n, $\Phi(n)$ を生成 *)

```
Remove[PrimeQ, ProvablePrimeQ];  
<<NumberTheory`PrimeQ`;  
Remove[w, p, q, n];
```

```
SeedRandom[]; (* 時刻をもとに擬似乱数を生成する *)  
w=512; (* 素数の桁数(二進数での) *)
```

```
While[True,  
  p=Random[Integer, {2^w / 2+1, 2^w-1}];  
  If[PrimeQ[p]&& ProvablePrimeQ[p], Break[]] (* 素数であるかどうか更に  
  詳細なテストを行う。しかしこれでも数学的には完璧ではない。 *)  
]
```

```
While[True,  
  q=Random[Integer, {2^w / 2+1, 2^w-1}];  
  If[PrimeQ[q]&& ProvablePrimeQ[q], Break[]] (* 素数であるかどうか更に  
  詳細なテストを行う。しかしこれでも数学的には完璧ではない。 *)  
]
```

```
n=p*q;  
phi[n]=(p-1)(q-1);
```

6 Wolfram Research, Inc. による数値計算スイート

7 Miller-Rabin テスト、ラビン・ミラーテスト、ミラーラビン法、などいろいろな呼称があるようだ。

8 PrimeQ および ProvablePrimeQ という確率的素数判定の function を備えている。この function は、ミラー・ラビンテストを含む複数の方法を組み合わせ実装されている。参照：<http://documents.wolfram.com/v5/AddonsLinks/StandardPackages/NumberTheory/PrimeQ.ja.html>

(* p,q, n, $\Phi(n)$ を表示 *)

p

q

n

phi[n]

754069940557657194249565172942891188563537280315440821728892356280
248926013462520667399102790533038532053539904112718485152427587393
0213149998648634029493

887809193851255023481293153646020467012718566911180493705100465683
858816731657615033140182181546513736430607210012330918423470182395
4904218950091173825449

669470226033957428553296504269024754711721999483798223101767298561
163163418029082378325086756538490827748098293773023732120992089302
323078365287899355742142560206460931836980697722522239860751904709
890086068177080661391201555511907189253034338101525743391942119769
67181823488445482501745674637237276299967357

669470226033957428553296504269024754711721999483798223101767298561
163163418029082378325086756538490827748098293773023732120992089302
323078365287899355741978372293020040615207611889863348695194279125
167423936633681379194790781237395175682980409604317788165093705058
25931329452686504803860557268288536492112416

(* e, eの逆元einv (授業ではdだった) を生成する *)

While[True,

e=Random[Integer,{0, n-1}];

If[GCD[e,n]==1 && GCD[e,phi[n]]==1, Break[]]

(* eがnと互いに素であれば, eはnを法とする剰余系 \mathbb{Z}_n スタ-の元である. さらに, e
が $\Phi(n)$ を法として逆元を持つ必要十分条件は $1=\text{GCD}(e,\Phi(n))$ である *)

]

{1,{einv,k}}=ExtendedGCD[e,phi[n]]; (* 拡張ユークリッド互除法を用いて,
 $einv * e + k * \Phi(n) = 1$ を満たすeinvを求める. これは $\Phi(n)$ を知らないと困難
である. *)

e

einv

Mod[e*einv+k*phi[n],phi[n]]==1 (* eとeinvが $\Phi(n)$ を法として逆元であるこ
とを確認 *)

108604088712909393721775106618919984020055286914702391363104356301
014238369807359109768319485161021527848700435437030027147288073177
321193202011997567524918507198749149185002837499257332596204929491
047979427480720200783103541658799907655760263612227555184064578952
49041672330204873490799353206161043928500225

261141794932509742716416548830959817233395125205426790467847981088
203007359612166178721270708972237418760980848314961697933025007261
731461040273815646305567876317609977216277404559113536608935874637
485135901995192357368364630870387091906770309619660197281051054028
85062307714882525458988733667037312573252737

True

(* 乱数の生成 *)

```
While[True,  
  m=Random[Integer,{1,n}];  
  If[GCD[m,n]==1, Break[]] (* mは剰余系Znスタ-の元 *)  
]
```

m

```
529623637477253362176849723624299442514150810728422881998779462845  
062821439199716785547947302448472247686870123772031353087150986094  
840357570622669422857983438938943123786669615668131346399885682530  
209615545211290016781246447524283541095740579854205370471554490885  
74613211788715690016515403301024930943666018
```

(* 乱数に対する署名の生成。これはeinvを知らないと困難である。 *)

```
s=PowerMod[m, einv, n]
```

```
33371844397272711741495770598159035855502186394553779733330875400764451  
455680343299805512827955711104407484519854370081806815464580757223532038  
976097836225499108081018677312298184247864870124317747312042146137073405  
946380425695109715674348249457282194712147631930518733004112384554389388  
800891465859237279138
```

(* 署名の検証。 Trueならば, 秘密鍵einv,p,qを持っている本人が生成したものと確信できる。 *)

```
m2=PowerMod[s, e, n]
```

```
m==m2
```

```
529623637477253362176849723624299442514150810728422881998779462845  
062821439199716785547947302448472247686870123772031353087150986094  
840357570622669422857983438938943123786669615668131346399885682530  
209615545211290016781246447524283541095740579854205370471554490885  
74613211788715690016515403301024930943666018
```

True

II.レポートシステムの認証方式の変更とその影響

RSA のプログラムに精神力と時間を使いすぎたので、勝手ながら省略させていただきます。と思ったのですが、さぼりつつも一応埋めます。申し訳ありません。

以下の方法は、どれか1つに全面的に移行するとなれば、デメリットが大きく響いてくるだろうと思われる。どれか1つを選ぶのではなく、現行のパスワードによる認証も含め、複数の認証をケースバイケースで選択・組み合わせられるようになることが理想である。

▶ RSA 署名を用いた認証方式を取り入れる場合

RSA 署名は、ほぼどのコンピュータでも利用でき、ユーザの教育さえ進めば非常に現実的な方法であろうと思われる。Microsoft Windows、Macintosh、Solaris や Linux、FreeBSD など、様々な環境が混在する SFC では、特にこの汎用性は重要であると思われる。ソフトウェアのみで実現できるので、商用のソフトウェアに加えて、フリーソフトウェア・オープンソースソフトウェアなどを組み合わせれば、ほぼ全てのコンピュータ環境に低い初期コストで認証を導入できるだろう。ただ、秘密鍵・失効証明書の管理などを全ユーザに求めるとなれば、ユーザの教育にコストがかかるであろう。

▶ IC カード認証方式を取り入れる場合

IC カードの場合、カードリーダーがある場所・カードリーダーが対応したコンピュータ環境からしか利用できないという問題がある。しかし、ユーザへの教育負荷は比較的低いと思われるのは大きなメリットである。しかし、レポートシステムの場合、自宅から提出できるということが非常に重要視されるだろうから、カードリーダーの購入をほぼ全学生に義務づけなければならない、という問題は大きなデメリットとなるだろう。初期コストも大きく、様々なコンピュータ環境に対応できるか、汎用性に疑問が残る。

▶ バイオメトリクス認証(指紋)を取り入れる場合

指紋認証の場合も、指紋認証装置がある場所・指紋認証装置が対応したコンピュータ環境からしか利用できないという問題がある。しかし、こちらもユーザへの教育負荷は比較的低いと思われるのは大きなメリットである。IC カードと違って、紛失の心配もない。しかし、IC カードによる認証と同様、レポートシステムに利用する場合、自宅から提出できるということが非常に重要視されるだろうから、指紋認証装置の購入をほぼ全学生に義務づけなければならない、という問題は大きなデメリットとなるだろう。初期コストも大きく、様々なコンピュータ環境に対応できるか、汎用性に疑問が残る。