署名データを組み込んだ RSA 公開鍵生成法の提案

北原基貴[†1]　西出隆志[†2]　櫻井幸一[†2]

公開鍵暗号において，一般的には公開鍵の正当性を保証する証明書が信頼できる認証局によって添付される．この仕組みを公開鍵基盤と呼ぶ．ここでは多くの場合公開鍵ディレクトリと呼ばれる，誰もが参照できる場所に公開鍵と証明書を保存する．公開鍵基盤の問題点として，公開鍵と証明書の両方を保存しなければならないという点がある．RSA 暗号では公開鍵に証明書を埋め込む手法が提案されているが．公開鍵の改ざんを行うことで所有者になりすましができるという問題点がある．本研究では既存の埋め込み手法を拡張し，改ざんが行われた場合それと検知できるシステムを作成し，公開鍵の改ざんを防ぐ手法を提案した．また，本研究ではこの方式を応用した匿名暗号も提案した．

Generating the RSA public key with the embedded signature

Motoki Kitahara,[†2] Takashi Nishide[†2] and Kouichi Sakurai[†2]

Usually, the certificate from a certification authority which shows the authenticity of a public key is added to the public key. This system is called Public Key Infrastructure (PKI). The problem of PKI, we must upload a public key and a certificate. In RSA cryptosystem, the technique to embed the certificate in a public key is proposed, but it has the problem that an attacker can impersonate by tampering the public key. We proposed a new method to prevent impersonations of a public key owner by extending technique. We can detect it if an attacker tampers. Furthermore, we proposed the anonymous encryption by applying this technique.

1. はじめに

公開鍵暗号の利用の際には公開鍵の正当性を検証する必要があるため，通常，その鍵が誰のものであるかを保証する証明書が信頼できる認証局によって添付される．公開鍵を受け取ったユーザはこの証明書を確認することで，その正当性を検証できる．このような機構のことを公開鍵基盤と呼ぶ．現在用いられている公開鍵基盤では，多くの場合公開鍵ディレクトリと呼ばれる，誰もが参照できる場所に公開鍵と証明書を保存する．

公開鍵基盤の問題点として，公開鍵ディレクトリに公開鍵と証明書の両方を保存しなければならず，保存領域を消費するという点がある．RSA 暗号では公開鍵に情報を埋め込む手法が提案されており[1)]，証明書を埋め込むことでこれを避けることができる．しかし，保存されている公開鍵に対して証明書の部分を参考に改ざんを行うことで，公開鍵を見ることの出来る人ならば誰でも，その公開鍵の所有者になりすましができるという問題点がある．証明書なしで正当性を検証できる暗号として，ユーザの ID を公開鍵として用いる ID ベース暗号がある[2)]．しかし現在知られている実現法では，鍵配布センターの秘密情報を使わなければ秘密鍵を作ることができないため，鍵配布センターがすべてのユーザの秘密鍵を知ってしまうというエスクロー問題がある[3)]．

本研究では既存の埋め込み手法を拡張し，公開鍵の改ざんを防ぐ手法を提案した．よく使われる RSA 公開鍵 $N = pq$ に対し，r を新たな検証用の値として導入し $N = pqr$ という三素数の積に拡張を行った．この r は鍵配布センターの公開鍵で暗号化し，証明書と一緒に公開鍵へ埋め込んだ．もし公開鍵の改ざんが行われると N の値が変わるため，鍵配布センターによって r が N の素因数かを検証することでなりすましの有無を判別可能となった．この手法では秘密鍵はユーザが作成するためエスクロー問題を解決できた．また，本研究ではこの方式を応用した匿名暗号も提案した．

2. 既存研究

2.1 公開鍵への情報の埋め込み

最も広く使われている RSA 暗号の応用例として，公開鍵 N の一部に予め決めておいた

†1 九州大学工学部電気情報工学科
Department of Electrical Engineering and Computer Science, Kyushu University
†2 九州大学システム情報科学府情報学専攻
Department of Imformatics, Kyushu University

既定値を埋めこませるというものがある．これを利用することで様々な情報を埋め込むことができる．

Canstone と Zuccherato が最初にこの考え方を実際のアルゴリズムを用いて提案した[1]．その後，Lenstra によって N の半分のビット長の値を既定でき，時間的にも効率的なものが提案された[4]．この手法では鍵生成の速度を RSA 暗号での鍵生成の速度とほぼ等しくすることができる．

Lenstra により効率的な手法が提案されたことにより，様々な応用例が提案されてきた．予め決めておく部分公開鍵配布センターの公開鍵で暗号化した秘密鍵を埋め込むことで，公開鍵配布センターのみに伝えるというエスクローを実現したり，公開鍵そのもののデータを少なくしたりすることが出来る手法が Joye により提案された[5]．その他にも，上位ビットを固定することにより画像から公開鍵を作り，それをまた画像に戻すという作業を行い，画像の見た目を変えずに画像自体を公開鍵として作成するという手法も Laih と Chen によって提案されている[6]．

2.2 Lenstra のアルゴリズム

Lenstra は公開鍵に効率的に既定値を埋めこませる事のできるアルゴリズムを提案した[4]．また，この論文中では ID の中の様々な部分に予め決めておいた値を埋め込む手法を提案している．ここでは，それらのアルゴリズムのうち，N の前半部分に入れ込む手法を紹介する．I のビットの長さを $\left|I\right|$ と表し，公開鍵 N に含ませたい値を I とする．

公開鍵 N に埋め込ませたい値を I とするとき，I のビットの長さを $\left|I\right|$ と表すとする．

(1) $N' = I \parallel R$ を計算する，ただし $\left|N'\right| = \left|I\right| + \left|R\right|$ を満たす．ここでの $\left|I\right| + \left|R\right|$ が想定する公開鍵のビット数の長さと等しくなる様に乱数 R を選ぶ．
(2) $\left|R\right|$ と同じ程度の大きさのランダムな素数 p を計算する．
(3) $Q' = \lceil N/p \rceil$ を計算する．
(4) $Q' + V$ の値が素数となる最小の V を探し，$q = Q' + V$ を計算する．
(5) $N = pq$ を求める．N の前半ビットが I と等しく, かつ $\left|N'\right| = \left|N\right|$ となっているか検証する．どちらも等しければ N を返す．満たしていなかった場合は (1) から繰り返す．

最終的に出力された N が既定値 I を含んだ RSA 暗号における公開鍵となる．$N = pqr$ に拡張する場合は (2) で p, r を生成し (3) で $Q' = \lceil N/pr \rceil$ を計算することにより求まる．

2.3 公開鍵基盤

公開鍵暗号では，公開鍵は誰でも見ることのできる鍵配布センターに預けられ，保存される．暗号化したいユーザがいた場合は自分で鍵配布センターから公開鍵を取ってきて，暗号化を行うこととなる．しかし基本的に公開鍵は秘密鍵をもとに計算されるため，公開鍵を見ただけでは誰のものなのかを知ることはできない．ユーザが鍵配布センターに公開鍵を預ける際に名前を一緒に登録したとしても，その名前を偽造することにより，他人に送られたはずの暗号文を復号できるという問題が発生してしまう．

この問題を解決するため，現代社会では公開鍵基盤が用いられている．この機構を最初に考えたのは MIT の学部 4 年の学生で，発表したのも卒業論文という形だった[7]．しかしこの機構は現実世界で使うには必須であり，その後急速に広まった．

この仕組としては次のようになる．公開鍵とユーザを一対一に対応させる保証を持たせた証明書が信頼できる認証局によって添付される．公開鍵を受け取ったユーザはこの証明書を確認することにより，正しいユーザの公開鍵かを検証する．実際に公開鍵暗号が使われる場面では，ユーザが作成した公開鍵に対し，ユーザの名前との対応付けが取れる証明書を認証局が作成する．ユーザはその証明書を受け取り，公開鍵と一緒に公開鍵配布センターに保存する．暗号化を行いたいユーザは公開鍵と証明書の両方を取ってきて，自分で証明書の正しさを検証し，公開鍵を用いて暗号化するという流れになる．

2.4 ID ベース暗号

ID ベース暗号とは，Shamir が 1984 年に初めて提案[2] したものである．この時はアイディアのみで，実際に提案したものは ID ベース署名のみである．メッセージを受け取りたいユーザは予め自分の ID に対応する秘密鍵を信頼できるセンターから受け取っておく．その後, 送り手は受け取り手の ID で暗号化し, 暗号文を送る．受取り手はセンターから受け取っている秘密鍵を用いて復号する．これが ID ベース暗号の考え方である．

現状よく知られている ID ベース暗号としては，ペアリングと呼ばれる楕円曲線上の点のみを群とする要素を用いた写像を用いるものと、平方剰余仮定を用いるものの 2 つがある．しかしどちらの手法でも鍵配布センターがユーザ全員の秘密鍵を知ってしまうという問題が発生する．

2.5 Self-certificate

1991 年に giraut が self-certified public keys という暗号を提案した[8]．この暗号では公開鍵自体が検証されており，別途証明書を用意する必要がないという利点がある．この暗号では秘密鍵はユーザが作り，その情報をもとに公開鍵配布センターが公開鍵を作成する．公開鍵配布センターだけが持つ情報を用いないと公開鍵を作成することはできないため，攻撃者は有効な公開鍵を作ることができない．認証を行わずに公開鍵を使うことができる．もし

攻撃者によって公開鍵を入れ替えられていたとしても，攻撃者が暗号文を復号できないため問題ないという考え方である．公開鍵が誰のものか検証したい場合は，ゼロ知識証明を利用して検証者とその公開鍵保持者の間で通信を行う．

3. 提 案 手 法

3.1 提案方式 (秘密鍵情報非埋め込み型)

3.1.1 要　　件

Lenstra の手法を用いたアルゴリズムでは自身の ID のみしか埋め込まれていなかったため，攻撃者による偽装という問題が発生した．他人の公開鍵を使って暗号化してしまった場合，目的の相手が復号できないだけでなく，公開鍵の所有者に復号されてしまう．このため公開鍵の所有者を確かめなければならない．ここでの要件は提案手法 (秘密鍵情報非埋め込み型) の要件に，公開鍵の偽造不可能性を加えたものとなる．以下にまとめたものを記す．

- 公開鍵証明書の添付は不要
- キーエスクロー問題はない
- 必要な鍵は増えない
- 正規のユーザ以外は公開鍵を作成できない

3.1.2 概　　要

ここでは，公開鍵の中に信頼できる認証局からの署名も埋め込むことを考える．公開鍵に誰もが検証可能な情報を埋め込み，これを検証することで公開鍵の信頼性を確かめることができる．また，秘密鍵はユーザ自身で作るため，公開鍵配布センターは暗号文を復号することはできないという利点がある．

図 1 に秘密鍵情報非埋め込み型の提案方式の概要を示す．数字で示した番号が準備段階，数字に’をつけたもので示した番号が公開鍵を受け取り，暗号化を行う段階である．

3.1.3 定　　義

- 公開鍵配布センターについて．
  - 公開鍵配布センターは送られてきた公開鍵 (N, e) を保持する．
- 認証局について
  本提案方式で書く認証局とは，以下の機能を持つものとする．
  - 認証はオフラインで行なうものとし，ID を持つ人が一意に判別できるとする．
  - ID と何らかのデータ x を受け取った際，正しい人物から ID を受け取っていれば $sig(x)$ を返す．

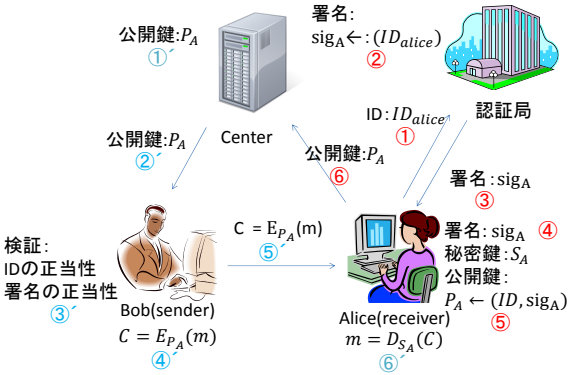

図 1 提案手法 (秘密鍵情報非埋め込み型) の略図
Fig. 1 proposal for $N = pq$

- 提案方式の登場人物について
  本提案方式では，以下 5 人の人物を考える．
  - アリス: 公開鍵を晒し，データを受け取るユーザ
  - ボブ: 公開鍵を受け取り，データを送るユーザ
  - チャーリー: 自分で作成した公開鍵をアリスの公開鍵だと偽りボブに使わせようとする攻撃者
  - 公開鍵配布センター: 上記で定義した，公開鍵を保存し認証を行う機関
  - 認証局: 上記で定義した，ユーザに対して証明書を送る機関

3.1.4 使用する暗号について

ID と認証局からの署名を公開鍵に埋め込むことのできる暗号であればどの暗号でもよい．ここでは，RSA 暗号と ElGamal 暗号に関して充足するかを確かめる．

- RSA 暗号の場合
  Lenstra アルゴリズムを用いることにより，公開鍵 N に好きな値を埋め込むことができる．よって $ID \parallel sig(ID)$ も公開鍵に埋め込むことができる．ここで埋め込んだ ID は誰でも見ることができ，署名の検証も行うことができる．ゆえに要件を満たす．
- ElGamal 暗号の場合

公開鍵の一つである素数 p をユーザ毎に分けることを考えると，素数は上位ビットを決めたとしても十分に選ぶ余地がある．よって $ID \parallel sig(ID)$ を公開鍵に埋め込むことができる．ここで埋め込んだ ID は誰でも見ることができ 署名の検証も行うことができる．ゆえに要件を満たす．

ここでは RSA 暗号を用いた手法とする．

3.1.5 提案方式 (秘密鍵情報非埋め込み型) の処理の流れ

(1) アリスは認証局に ID_{Alice} を送り，証明書 $sig(ID_{Alice})$ を受け取る．

(2) アリスは Lenstra のアルゴリズムを用いて $N = pq$ という 2 つの素数に対しての N を作成する．N の前半ビットは $H(ID_{Alice}) \parallel sig(ID_{Alice})$ と等しくなるように作成する．また，$\gcd(e, (p-1)(q-1)) = 1$ となる e，$ed \equiv 1 \bmod (p-1)(q-1)$ となる d を求める．

(3) アリスは作成した (N, e) を公開鍵配布センターに渡す．公開鍵配布センターは認証なしで (N, e) を受け取る．(悪意あるユーザが適当な (N, e) を渡すことも考慮してよい)

(4) ボブは $H(ID_{Alice})$ を計算し，公開鍵配布センターに保存されている公開鍵 N の前半部分と等しくなっている公開鍵 (N, e) の組を受けとる．

(5) ボブは受け取った公開鍵 N に関して，$sig(ID_{Alice})$ の値が正しい ID_{Alice} から作られたものかを検証する．間違っていれば棄却する．

(6) ボブは (N, e) を使い，RSA 暗号で平文を暗号化し，受信者に送る．

(7) アリスは自分の持つ秘密鍵 d で復号し，平文を得る．

3.1.6 安全性評価

- チャーリーが自分でアリスの証明書を作ることを考える．
  認証局では認証があるため，認証局に ID_{Alice} を送り，証明書 $sig(ID_{Alice})$ を受け取るという作業はできない．ここでできるのは自分の $ID_{Charlie}$ を用いて証明書 $(ID_{Charlie})$ を受け取ることのみである．これを用いて暗号化を行うと，署名部にはアリスではなくチャーリーの ID が含まれてしまう，公開鍵配布センターはアリスの公開鍵ではないということはすぐにわかる．よってなりすましを行うことはできない．
- チャーリーがアリスの用いた証明書を流用することを考える．
  $N = pq$ の場合，アリスが一度用いた既知の証明書 $sig(ID_{Alice})$ は誰でも用いることができる．ここでの認証は署名部のみを用いて行なうため，公開鍵 N の下位ビットに関しては確認しない．よって既知の証明書を流用した場合でも検証では正しい公開鍵だと判断を下すことになる．

この問題を解決する手段として，公開鍵配布センターは一度きた署名部が同じ値になっている公開鍵を送られてきた場合，棄却するという手法がある．署名を認証局から得ることができるのは正しい ID を持つアリスだけであるため，この前提であればなりすましは行えないといえる．

この手法の欠点として，公開鍵を送られてきた際に自分の持つ全ての公開鍵を調べ，一致するものがないか確認を行わなければならないということがあげられる．また，公開鍵配布センターが複数個存在するとした場合，全ての公開鍵配布センターは同期されていなければならず，保存されている公開鍵をネット等を通じて調べる必要がある．これらの問題があるため，全ての鍵を調べる必要がない，公開鍵だけを見て認証が可能な秘密鍵情報埋め込み型の方式を提案する．

3.2 提案方式 (秘密鍵情報埋め込み型)

3.2.1 要　　件

秘密鍵情報非埋め込み型では，公開鍵の証明書を使い改ざんができるという問題があった．この手法を防ぐ条件について考える．署名の流用を行ったとしても有意な公開鍵を作成することができないという条件が最適だが，正規のユーザ以外は公開鍵を作成できず，作成された公開鍵は署名の流用を行うことができないという条件でも同程度の安全性を持つため，こちらでもよいとする．後者の場合の要件は以下となる．

- 公開鍵証明書の添付は不要．
- キーエスクロー問題はない．
- 必要な鍵は増えない．
- 正規のユーザ以外は公開鍵を作成できない．
- 署名の流用はできない．

3.2.2 概　　要

公開鍵に証明書と秘密鍵情報の一部を埋め込むことで公開鍵の改ざんを防ぐ手法を提案する．秘密鍵情報埋め込み型の提案方式においても公開鍵に認証局の署名を埋め込む．こちらでは署名の流用を防ぐため，より強固な認証を行う．今回はよく使われる RSA 公開鍵 $N = pq$ に対し，r を新たな検証用の値として導入し $N = pqr$ という三素数の積に拡張した．この r は鍵配布センターの公開鍵で暗号化して公開鍵へ埋め込んだ．公開鍵の改ざんが行われると N の値が変わるため，鍵配布センターによって r が N の素因数かを検証することでなりすましの有無を判別可能となった．この手法では秘密鍵はユーザが作成するためエスクロー問題を解決できた．

図**図 2** に秘密鍵情報埋め込み型の提案方式の概要を示す．数字で示した番号が準備段階，数字に’ をつけたもので示した番号が公開鍵を受け取り，暗号化を行う段階である．

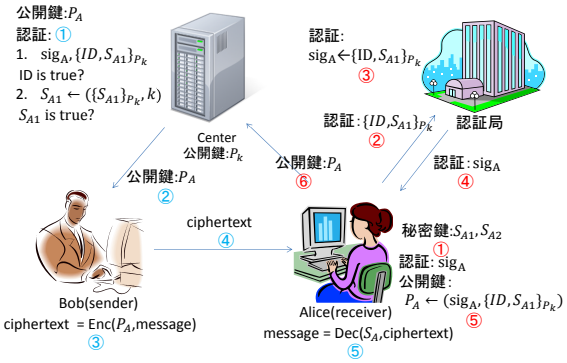

図 2 提案手法 (秘密鍵情報埋め込み型) の略図
Fig. 2 proposal for $N = pqr$

3.2.3 定　　義

- 暗号化の表記
  提案方式 (秘密情報埋め込み型) では，$E_k(M)$ と書いた場合，公開鍵暗号を用いて鍵 k でメッセージ M を暗号化することを意味する．
  Lenstra の手法では使用する暗号は RSA であった．しかし，本提案方式では秘密情報の一部を公開鍵配布センターが知っておく必要がある．RSA 暗号の場合，秘密情報である公開鍵の素因数が一つでも漏れていまうと，そのまま秘密鍵までわかってしまうため，そのまま使うことはできない．RSA を拡張した暗号が必要となる．
- RSA 暗号の拡張
  今回は，RSA 暗号に用いる公開鍵を，$N = pqr$ とする手法について考える．この場合であっても， $ed \equiv 1 \bmod (p-1)(q-1)(r-1)$ とを満たす d を秘密鍵，$\gcd(e, (p-1)(q-1)(r-1)) = 1$ を満たす e と $N = pqr$ となる N を公開鍵とすることで RSA 暗号は成り立つ．安全性としては p, q, r のうち一つが漏れてしまったとしても，残り二つの素数を求める問題は RSA 暗号における素因数分解問題と同等程度に難しいとすると，秘密鍵 d そのものが漏れてしまう心配はない．
- 公開鍵配布センター
  提案手法 (秘密鍵情報埋め込み型) における公開鍵配布センターとは，以下の機能を持つものとする．
  (1) 公開鍵配布センターは自身の公開鍵 P_{center} と，それに対応する秘密鍵 S_{center} を持つ．
  (2) 公開鍵配布センターはユーザから公開鍵 (N, e) を受け取った場合，それを保持する．
  (3) 公開鍵配布センターはユーザから公開鍵の取得を求められた場合，まず ID を受け取る．その後受け取った ID から $H(ID)$ を求め，この値とにおける先頭部分が等しくなっている公開鍵 N を探す．
  (4) 同じ値があった場合，その公開鍵に対して認証を行う．公開鍵 N に保存されている $E_{P_{center}}(r)$ に対して秘密鍵 S_{center} を用いて r を取り出し，公開鍵 N がこの値 r で割り切れるか検証する．また，署名部 $sig(ID)$ が ID の署名になっているか検証する．
  (5) 上記が満たされた場合，公開鍵 (N, e) を送る．
- 認証局について
  提案手法 (秘密鍵情報非埋め込み型) と同様とする．
- 提案方式の登場人物について
  提案手法 (秘密鍵情報非埋め込み型) と同様とする．

3.2.4 使用する暗号について

ID と認証局からの署名を公開鍵に埋め込むことのできる暗号であればどの暗号でもよい．ここでは，RSA 暗号と ElGamal 暗号に関して充足するかを確かめる．

- RSA 暗号の場合
  Lenstra アルゴリズムを用いることにより，ユーザが秘密鍵，公開鍵を作成しつつ公開鍵 N に好きな値を埋め込むことができる．よって $ID \parallel sig(ID)$ も公開鍵に埋め込むことができる．また，$N = pqr$ と公開鍵を変形させ，r の情報を埋め込むことを考える．攻撃者により公開鍵 N の後半部分が改ざんされてしまった場合，r が N の素因数ではなくなる．このことより署名の流用は行うことができないといえる．ただしこの r が攻撃者に知られてしまうと，r を素因数として用いて N を作成できてしまうため，

公開鍵配布センターの公開鍵により暗号化して埋め込む．この場合，要件をすべて満たすといえる．

- ElGamal 暗号の場合
  公開鍵 $g^x \bmod p$ に値を含ませることは難しく，p に値を含ませた場合偽造が容易である．このため，こちらの手法では難しい．

今回は RSA 暗号を用いた手法とする．

3.2.5 提案方式 (秘密鍵情報埋め込み型) の処理の流れ

(1) アリスは認証局に ID_{Alice} を送り，証明書 $sig(ID_{Alice})$ を受け取る．
(2) アリスは素数 r を生成する．
(3) アリスは素数 r を公開鍵配布センターの公開鍵 P_{center} で暗号化する．
(4) アリスは Lenstra のアルゴリズムの拡張版を用いて $N = pqr$ という 3 つの素数に対しての N を作成する．N の前半ビットは $H(ID_{Alice}) \parallel sig(ID_{Alice}) \parallel E_{P_{center}}(r)$ と等しくなるように作成する．また，$\gcd(e, (p-1)(q-1)(r-1)) = 1$ となる e, $ed \equiv 1 \bmod (p-1)(q-1)(r-1)$ となる d を求める．
(5) アリスは作成した (N, e) を公開鍵配布センターに渡す．公開鍵配布センターは認証なしで (N, e) を受け取る．(悪意あるユーザが適当な (N, e) を渡すことも考慮してよい)
(6) ボブはアリスの ID_{Alice} を公開鍵配布センターに送る．
(7) 公開鍵配布センターは $H(ID_{ALice})$ と等しい公開鍵に対する検証を行い，正しければ公開鍵 (N, e) をボブに送る．
(8) ボブは (N, e) を使い，RSA 暗号で平文を暗号化し，受信者に送る．
(9) アリスは自分の持つ秘密鍵 d で復号し，平文を得る．

3.2.6 安全性についての検証

- 攻撃者による公開鍵の偽装
  - チャーリーが自分でアリスの証明書を作ることを考える．
    認証局では認証があるため，認証局に ID_{Alice} を送り，証明書 $sig(ID_{Alice}, H(r))$ を受け取るという作業はできない．ここでできるのは自分の $ID_{Charlie}$ を用いて署名 $(ID_{Charlie})$ を受け取ることのみである．これを用いて暗号化を行うと，署名部にはアリスではなくチャーリーの ID が含まれてしまう，公開鍵配布センターはアリスの公開鍵ではないということはすぐにわかる．よってなりすましを行うことはできない．
  - チャーリーがアリスの用いた証明書を流用することを考える．
    $N = pqr$ の場合，アリスが一度用いた既知の証明書 $sig(ID_{Alice})$ は誰でも用いることができる．また，$E_{P_{center}}(r)$ も見ることができ，これらをもとにした公開鍵は作ることができる．しかし，チャーリーは公開鍵配布センターの秘密鍵を知らず，$E_{P_{center}}(r)$ を復号することはできない．そのため，r の値を知ることができず，公開鍵配布センターが N を r で割り切れるかの検証を行う際，r で割り切れる N を生成するのは難しいといえる．
- 公開鍵配布センターに対する安全性
  公開鍵配布センターに対する安全性，すなわち鍵供託問題について考える．$N = pqr$ の場合の提案方式では，公開鍵配布センターは認証を確認しており，$N = pqr$ の r を知ることができる．これを用いて，公開鍵 N が r で割り切れる事が出来るか，ということを証明書の検証に利用しているためである．この秘密情報の一部を知ることによりその他の秘密情報を知ることができるか検証する．
  公開鍵配布センターは $N = pqr$ のうちの r が既知であり，$ed_r \equiv 1 \bmod r-1$ となる d_r を求めることで，$M' = c^{d_r} \bmod r$ の値は計算することができる．ここで求めた M' が求めたい平文 M と等しくなる，すなわち $M = M'$ となる場合は，$0 < M \bmod pqr < r$ を満たすときのみである．まず明らかにこの式が成り立つのが，M の値が r より小さくなっている場合である．これを防ぐため送りたいメッセージはある程度大きくする必要がある．ボブは r の値を知ることはできないため，十分に r より大きくなると想定される $\sqrt{N}$ 程度の大きさ以上のメッセージを送ればよいと考えられる．M がある程度以上大きい場合の安全性は，r を知らない場合の $1/2^{pqr}$ から $1/2^{qr}$ 程度まで落ちてしまう．しかしこの場合でも一般的な RSA 暗号と同程度のセキュリティレベルを持つ．ゆえに鍵供託問題は存在しないといえる．

3.3 考　　察

3.3.1 匿名化手法

誰の公開鍵かわからないことを利用し，匿名通信に利用できる．匿名通信とは，送った情報が第三者にわからないだけでなく，誰が誰に送ったかを秘匿する通信のことである．匿名暗号は 2006 年に ID ベース暗号を改善し，ID を用いて暗号化した暗号文から ID を漏れないようにした方式が，Boyen と Waters によって提案されている[9]．この暗号文を利用し，誰と誰が通信を行ったかもわからないようにする方式が匿名暗号通信である．実際に通信が行われる時は以下の流れとなる..

(1) 送信者は匿名暗号を用いて，匿名性のある暗号文を作成する．
(2) 送信者は誰もが見ることの出来るサーバにこの暗号文を置く．
(3) サーバは多くの人からこの匿名暗号文を受け取る．
(4) 受信者はサーバに保存されている暗号文を全て受信する．
(5) 受信者は自分の持つ秘密鍵で全ての暗号文に対し復号を行い，復号できた暗号文を自分に対して送られたものだと判断し，平文を得る．

この手法では自分に対して送られた暗号文がどれなのかを，受信者でも知ることができない．そのため上記のように全ての暗号文に対して復号処理を行うしかなく，処理に時間がかかってしまうという欠点がある．そこでより効率的に自分宛の暗号文がどれなのかを判断する手法を考える．

本提案手法では，暗号文に匿名性が存在するがこのままでは効率化ができない．そのため，公開鍵に匿名性を保持させ，暗号文と一緒に保存することを考える．この提案手法では公開鍵に ID も埋め込んでおり，署名も ID に対してそのまま行なっているため，公開鍵自体に匿名性はない．しかし埋め込み方を多少変化させると公開鍵に匿名性を持たせることができる．この匿名化させた公開鍵を暗号文と一緒に保存し，検索を容易にする．

(1) 受信者は匿名性公開鍵を作成し，自身が決めたパスワードと共に鍵配布センターに預ける
(2) 送信者は受信者の決めたパスワードを鍵配布センターに伝え，受信者の公開鍵を得る．
(3) 送信者は匿名性公開鍵を用いて暗号化を行う．
(4) 送信者は誰もが見ることの出来るサーバに暗号文と匿名性公開鍵の組を保存する．
(5) サーバは多くの人からこの暗号文と匿名性公開鍵の組を受け取る．
(6) 受信者はサーバに保存されている暗号文と匿名性公開鍵の組を全て受け取る．
(7) 受信者は自分の公開鍵と等しいものを探し，その公開鍵と組になっている暗号文に対して復号を行う．
(8) 受信者は平文を得る．

この手法の場合，受信者は自分の公開鍵と等しいものを探してくればいいので，復号処理を行う必要がなく，また探索時間もソートを行うことにより log のオーダで行うことができる．注意点としては，匿名性公開鍵も一緒に保存してしまっているため，この公開鍵を持つ人であれば暗号文が誰に送られているか判別で来てしまうという点があげられる．このため完全な匿名性とは言えず，限定的な匿名性となる．

3.3.2 既存手法との比較

ここでは主に秘密鍵情報埋込み型の手法と，一般的と思われる暗号に対して比較を行う．

- 公開鍵基盤との比較
  提案方式と比較した場合，証明書を公開鍵と添付させなければならないため，通信路や管理の問題がある．提案方式では公開鍵のみで暗号化できるが，鍵配布センターを信頼しなければならないという欠点がある．
- ID ベース暗号との比較
  提案手法 (秘密情報埋め込み型) と比較すると，相手の ID がそのまま公開鍵となっているため公開鍵配布センターに公開鍵を取りに行く必要がないという利点がある．また秘密鍵配布の際にはユーザの認証が必要になるが，一度全員に秘密鍵を配ってしまえば鍵配布センターの存在そのものが必要なくなるという利点もある．この手法の欠点として，鍵配布センターが全員分の秘密鍵も配布するため，鍵配布センターが秘密鍵を知ってしまい暗号文の復号ができるという鍵供託問題が存在することがあげられる．
- Self-certified public keys との比較
  提案方式では公開鍵配布センターが認証を行うため，公開鍵を使うユーザからすると公開鍵 “certified” とみなすことができ，どちらも別に証明書を用意する必要がないという点でこの手法と同じメリットを持つといえる．提案方式との違いはこの手法では公開鍵をそのまま暗号化に使うのではなく，ユーザの ID も用いて暗号化するという点と公開鍵を作成するのがユーザかセンターかという違いがある．また，提案方式では幾つかの前提が必要となるが self-certified では公開鍵基盤と同程度の前提で作成することができる．

表 1 提案方式と他の手法の比較表
Table 1 Comparison with Existing Methods

	公開鍵基盤	ID ベース暗号	Self-certified	提案手法
キーエスクロー問題	なし	あり	なし	なし
送信者の公開鍵取得	必須	不要	必須	必須
公開鍵証明書作成	必須	不要	不要	必須
公開鍵証明書添付	必要	不要	ID が必要	不要
公開鍵の検証	必須	不要	不要	必須
公開鍵の検証者	誰でも可能	誰でも可能	誰でも可能	センターのみ
公開鍵の秘匿性	なし	なし	あり	あり

3.3.3 秘密鍵情報非埋込み型と秘密鍵情報埋込み型の比較

秘密鍵情報非埋め込み型の場合，既定値が 560 ビットで済む，これを Lenstra のアルゴリズムを用い，RSA 暗号の公開鍵 N を作成した場合，法が 1024 ビットとなり，140 ビットとした楕円曲線暗号と比較しても，無駄がない形といえる[10]．しかし，公開鍵配布センターでの認証を行なっていないため，署名の流用という問題が発生してしまう．こちらで実装を考えた場合,．二度目に来た署名は流用が行われたとして棄却する必要がある．これにはすべての公開鍵をチェックしなければならないため，即時性が失われることに加え，公開鍵配布センターが複数個ある場合は完全な同期をとり常にアクセス出来る状況になければならない．

秘密鍵情報埋め込み型の場合，しかし前述した通り，1200 ビットの p, q と 512 ビットの r が必要になるという欠点がある．p, q, r の長さを均等にすることができないため，RSA の N の法を 3000 ビット取ったとしても，素数 r の大きさが 512 ビットであるため，この分の安全性しか持てないという問題がある．

4. おわりに

本論文では公開鍵のみで ID 把握，ユーザの認証，暗号化の行える公開鍵暗号を提案した．この手法のメリットとしては

- 公開鍵配布センターがユーザの秘密鍵を知ることはない (キーエスクロー問題が存在しない)
- 公開鍵のみでユーザの判別，認証，暗号化を行うことができる
- 別途証明書を付ける必要がない

という点があげられる

また，今後の課題として

- 一般的な公開鍵に適用
- 実装上の効率の低下について調査
- よりよい形での認証の埋め込み方について
- 安全性証明

を考えている．安全性証明に関しては $N = pqr$ の RSA 暗号としているが，この場合に p が漏れてしまっても本当に安全と言い切れるのかについて検証を行う必要がある．実装上の効率低下についての調査はアルゴリズム単位でのみ行っただけなため，提案方式をシミュレートした際に致命的な遅れがでないかを実装上で検証する．

謝辞 本研究の一部において, 第二著者は，日本学術振興会 科学研究費補助金 若手研究 B (23700021)，また第三著者は挑戦的萌芽研究 (23650008) による補助を受けている．

参 考 文 献

1) Vanstone, S. and Zuccherato, R.: Short RSA keys and their generation, *Journal of Cryptology*, Vol.8, No.2, pp.101–114 (1995).
2) Shamir, A.: Identity-based cryptosystems and signature schemes, *Advances in cryptology*, Springer, pp.47–53 (1985).
3) Boneh, D. and Franklin, M.: Identity-based encryption from the Weil pairing, *Advances in Cryptology?CRYPTO 2001*, Springer, pp.213–229 (2001).
4) Lenstra, A.: Generating RSA moduli with a predetermined portion, *Advances in Cryptology-Asiacrypt' 98*, Springer, pp.1–10 (1998).
5) Joye, M.: RSA moduli with a predetermined portion: Techniques and applications, *Proceedings of the 4th international conference on Information security practice and experience*, Springer-Verlag, pp.116–130 (2008).
6) Laih, C. and Chen, K.: Generating visible RSA public keys for PKI, *International Journal of Information Security*, Vol.2, No.2, pp.103–109 (2004).
7) Kohnfelder, L.M.: Toward a Practical Public-Key Cryptosystem, Bachelor Thesis, Massachusetts Institute of Technology (1978).
8) Girault, M.: Self-certified public keys, *Proceedings of the 10th annual international conference on Theory and application of cryptographic techniques*, Springer-Verlag, pp.490–497 (1991).
9) Boyen, X. and Waters, B.: Anonymous hierarchical identity-based encryption (without random oracles), *Advances in Cryptology-CRYPTO 2006*, Vol.4117, pp. 290–307 (2006).
10) 下山武司，伊豆哲也，小暮 淳，安田雅哉：楕円曲線暗号と RSA 暗号の安全性比較，暗号と情報セキュリティシンポジウム (2010).