ElGamal型協力署名の構成と鍵管理・鍵寄託システムへの応用

宮崎真悟[†] 石本関[††,☆]
新保淳[†††] 櫻井幸一[†]

コンピュータシステムYakshaは，クライアントの身元を保証するシステムであるKerberosにディジタル署名機能を付加したものである．Yakshaでは RSA 方式を用いていたが，本稿では離散対数ベースの公開鍵暗号であるElGamal方式を利用したYakshaの一形式を示す．通常，ElGamal署名では RSA 署名と違いメッセージ回復が行われない．そこで今回は新たにメッセージ回復型のElGamal協力署名を提案する．また従来のYakshaにおける鍵寄託のシステムでは，セッション鍵とそれを利用するユーザとの関係がサーバ側にすべて知られていた．今回発表する鍵寄託システムでは，セッション鍵を扱う機関をサーバとは別に用意し，鍵とユーザとを結び付ける情報を分割することでユーザのプライバシーの保護を図る．

Jointsignatures Based on the ElGamal Scheme and Its application to Key Management/Escrow Systems

SHINGO MIYAZAKI,[†] KAN ISIMOTO,[††,☆] ATSUSHI SHINBO[†††] and KOUICHI SAKURAI[†]

The Yaksha is a computer system with the digital signature mechanism over the Kerberos system for authenticating clients. While the Yaksha system uses the RSA public-key cryptosystem, we show a form of the Yaksha with the ElGamal cryptosystem based on the discrete logarithm problem. Note that the original ElGamal signature do not have message recovery feature unlike RSA. Then, we propose a new ElGamal-type jointsignature scheme with message recovery. We further give a new key escrow system on the Yaksha for protecting user's privacy with separating information linked a session key to its users, whereas, in the key escrow system on the Yaksha, the server knows both a session key and its users.

1. はじめに

インターネットが世界に急速に普及しつつある現在，コンピュータセキュリティを扱う技術や提案が数多くなされている．セキュリティシステムのパイオニア的存在であるKerberos[5]もまたそれらのうちの1つに数えられる．Kerberosは，チケットと呼ばれる暗号化情報を用いてクライアントの身元を保証するシステムだが，機能の拡張を考えるにつれいくつかの問題が生じてきた．それらの原因はKerberosシステムが共通鍵暗号のみを用いて実現されていたことにあった．

そのKerberosに公開鍵暗号を用いて問題となっていた点を解決したのがセキュリティシステムYaksha[1]である．Yakshaでは公開鍵暗号の秘密鍵を2者（クライアントとサーバ）に分割して持たせることにより，片方のみによる署名と両者の協力署名[7]とを可能にしている．ここでの片方のみによる署名は他方への暗号文という意味を持ち，受け取った側がその署名にさらに署名することで作成される両者の協力署名が従来のディジタル署名としての役割を果たす．Yakshaではこれらの署名を応用して，秘密通信および相手の認証を行っている．またYakshaはディジタル署名やe-mailなど既存のシステムの組込みや近年注目を集めている鍵寄託を睨んだ設計がなされている．

従来YakshaはRSA方式を用いていたが，RSA方

† 九州大学大学院システム情報科学研究科情報工学専攻
Department of Computer Science and Communication Engineering, Kyushu University
†† 九州大学工学部情報工学科
Department of Computer Science and Communication Engineering, Kyushu University
††† 株式会社東芝研究開発センター情報・通信システム研究所
Research and Development Center, Toshiba Corporation
☆ 現在，ブラザー工業株式会社
Presently with Brother Industries

Signer1		Signer2
$r_1 = g^{-k_1} \bmod p$		
$C = (c_1, c_2) = [M]_{yg^{-x_1}}$	$\xrightarrow{C, r_1}$	
		$M = c_2 / {c_1}^{x_2} \bmod p$
		$r_{12} = M \cdot r_1 \cdot g^{-k_2} \bmod p$
		$s'_2 = k_2 - x_2 r_{12} \bmod (p-1)$
$s'_1 = k_1 - x_1 r_{12} \bmod (p-1)$	$\xleftarrow{r_{12}, s'_2}$	
$s_{12} = s'_1 + s'_2 \bmod (p-1)$		

図1 協力署名プロトコル
Fig. 1 Protocol for the joint signatures.

式とともに公開鍵暗号を代表する離散対数ベースの暗号法での検討も有用であるとの判断から，今回我々は離散対数ベースの暗号方式の代表として ElGamal 方式による Yaksha を検討した．通常の ElGamal 署名ではメッセージの回復をサポートしていないなどの RSA 署名との相違点のため，従来のシステムを直接的には適用できない．その点を解決するために，離散対数ベースのメッセージ回復型署名[3]を 2 者で行うメッセージ回復型の ElGamal 協力署名を提案する．

またあわせて Yaksha の鍵寄託方式の改良を検討した．これまでに述べられている Yaksha での鍵寄託方式ではセッション鍵の生成・発行・保管のすべてを Yaksha の認証サーバが担っていたが，これでは認証サーバにはクライアント間のセッション鍵の関係をすべて知ることができる．これにより認証サーバには捜査機関がだれを参考人として捜査しているのかを特定したり，クライアント間の通信を盗聴したりすることが可能となる．

Yaksha の鍵寄託方式におけるプライバシー保護の強化を目的としたシステムを示す．ここではセッション鍵の生成・保管を行う機関をそれぞれ用意し，またブラインド復号化[12]という概念を利用することで，セッション鍵とユーザとを結び付ける情報を分散している．

2. ElGamal 方式による巡回型協力署名

ここでは今回検討した ElGamal 協力署名を作成するプロトコルを示す（**図1** 参照）．その前に通常の ElGamal 方式について簡単に説明しておく．まず，用いるパラメータは以下のとおり．

p ： 素数
q ： $p-1$ の因数である素数
g ： p を法とした 1 の q 乗根

ElGamal 方式の秘密鍵は x，公開鍵は (y, g, p) で表され，これらの関係式は以下のようになる．

$$y = g^x \bmod p$$

またメッセージ M に対する通常の ElGamal 方式での暗号化・復号化操作および署名の作成については以下のとおり．

- 暗号化操作
$$C = (c_1, c_2) = (g^k \bmod p, my^k \bmod p) \quad (k : \text{乱数})$$
- 復号化操作
$$m = c_2 / c_1^x \bmod p$$
- 署名認証子
$$(r, s) = (g^k \bmod p, \; k^{-1}(H(m) - xr) \bmod (p-1))$$
- 認証式
$$g^{H(m)} = y^r r^s \bmod p$$

今回提案する ElGamal 協力署名は，2 者で 1 つの署名を作成することを目的とするものである．具体的には秘密鍵を 2 つに分割し，それを 2 者が分担して保持する，という手段を用いる．以下に鍵をどのように分割するかを示す．

$$y \equiv g^{x_1 + x_2} \bmod p \tag{1}$$

ここで，x_1，x_2 は秘密鍵，(y, g, p) は公開鍵である．ここに公開鍵 (y, g, p) は秘密鍵 $x_1 + x_2$ にのみ対応していることに注意する．秘密鍵 x_1，x_2 は協力署名を行う 2 者が別々に保管し，公開鍵 (y, g, p) はだれもが知りうるものとする．また第 3 者による署名の偽造を防ぐため，g^{x_1}，g^{x_2} もそれぞれ外部に秘密にしておく．鍵の対応について，以下に示す．

秘密鍵 x_1 ↔ 公開鍵 (yg^{-x_2}, g, p)
秘密鍵 x_2 ↔ 公開鍵 (yg^{-x_1}, g, p)
秘密鍵 x_1，x_2 ↔ 公開鍵 (y, g, p)

このように秘密鍵を 2 つに分けることで，鍵のペアが 3 種類できることが分かる．つまり，

1. 秘密鍵 x_1 のみによる署名
（確認には x_2 と公開鍵が必要）
2. 秘密鍵 x_2 のみによる署名
（確認には x_1 と公開鍵が必要）

3. 秘密鍵 x_1, x_2 の協力署名
（確認には公開鍵が必要）

という3通りの署名が作成可能となる．

また後述のYakshaへの組込みを考慮して，今回はメッセージ回復型のElGamal署名について検討する．メッセージ回復型のElGamal署名および認証式（メッセージ回復操作）は以下のとおり．

- 署名認証子
  $(r, s) = (M \cdot g^{-k} \bmod p,$
  $k - r'x \bmod (p-1))$
  （ただし $r' = r \bmod (p-1)$）
- 認証式
  $M = g^s y^{r'} r \bmod p$

ここに署名作成について次の2つのケースに分けて検討する．

(1) 署名者1から署名者2への署名，署名者2から署名者1への署名

(2) だれもが確認できる署名

これら2種類の署名生成アルゴリズムについて説明する．

まずはじめに(1)の署名は，アルゴリズム的に暗号化操作と同様にして作成する．厳密にいえば，署名ではなく署名の意味を持つ暗号化である．たとえば，署名者1の持つ公開鍵 yg^{-x_1} で暗号化されたメッセージは，署名者2の所有する秘密鍵 x_2 でしか復号できない．逆に署名者2から見れば，秘密鍵 x_2 で復号できる暗号文を生成する公開鍵は署名者1のみが所有するものであるから，暗号文の作成者が署名者1であることが分かる．

次に(2)の署名だが，これは前述のメッセージ回復型のElGamal署名の作成と同様に行う．この署名は署名者1と署名者2の協力署名であるため，署名認証子 (r_{12}, s_{12}) は

$$r_{12} = M \cdot (g^{-k_1}) \cdot (g^{-k_2}) \bmod p$$
$$s_{12} = (k_1 - r'_{12}x_1) + (k_2 - r'_{12}x_2) \bmod (p-1)$$

のように計算する．この場合，$s'_i = k_i - r'_{12}x_i \bmod p$ の作成にはあらかじめ r_{12} を知っておく必要があるため，署名の作成は以下のように行う．署名者1を S_1，署名者2を S_2 とする．

(1) S_1 は乱数 k_1 を用意して

$$r_1 = g^{-k_1} \bmod p$$

を作成し，秘密鍵 x_2 に対応する公開鍵である yg^{x_1} を用いて暗号化したメッセージ $C = (c_1, c_2)$

$$c_1 = g^k \bmod p$$
$$c_2 = M \cdot yg^{-x_1} \bmod p$$

を r_1 とともに S_2 に送る．

(2) S_2 は

$$M = c_2 / c_1^{x_2} \bmod p$$

として S_1 から送られてきた暗号文を復号（確認）し，取り出した M と r_1 および乱数 k_2 から

$$r_{12} = M \cdot r_1 \cdot g^{-k_2} \bmod p$$
$$s'_2 = k_2 - x_2 r_{12} \bmod (p-1)$$

を作成してそれらを S_1 に送る．

(3) S_1 は入手した r_{12}, s'_2 から

$$s'_1 = k_1 - x_1 r_{12} \bmod (p-1)$$
$$s_{12} = s'_1 + s'_2 \bmod (p-1)$$

を作成し，

$$M = g^{s_{12}} y^{r'_{12}} r_{12} \bmod p$$

として署名を確認してから，(r_{12}, s_{12}) を要求する人物へ送信する．

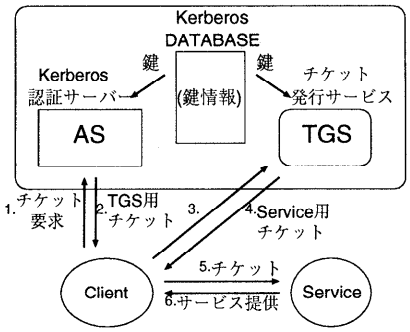
こうして作成された署名は，公開鍵 y を所有するだれもが認証できる．

3. Kerberos

Kerberosはサービスに対してクライアントの身元を証明する認証方式である．認証はチケットと呼ばれる暗号化情報を用いて行われ，その内容は，チケットが有効である2者（クライアントとサーバ）の識別子 c, s，それら2者間で用いる秘密鍵 $K_{c,s}$，および送信ユーザのアドレス $addr$，タイムスタンプ ts，チケット有効期限 $time\text{-}exp$ という6つからなる．Kerberosではいずれの秘密通信の際にも共通鍵暗号（DES）のみを用いている．

Kerberos（version5[5]）の一連の動作について**図2**に示す．ここで，クライアントをC，Kerberosの認証サーバ（Authentication Server）をAS，チケット発行サービス（Ticket Granting Service）をTGS，クライアントの要求するサービスをSで表す．また鍵は K，チケットは T で表し，添字でどの2者間で有効であるかを示している．

しかし，Kerberosはクライアントと各種サービス間でのユーザ認証に主眼をおいてあるため，クライアント間の通信にまで拡張した場合に暗号文（署名）の作成者の証明（署名者認証）ができないという問題点



1.	C	$\longrightarrow$	AS	(c, tgs)
2.	AS	$\longrightarrow$	C	$([K_{C,TGS}, [T_{C,TGS}]_{K_{TGS}}]_{K_C})$
3.	C	$\longrightarrow$	TGS	$([A_C]_{K_{C,TGS}}, [T_{C,TGS}]_{K_{TGS}}, s)$
4.	TGS	$\longrightarrow$	C	$([K_{C,S}, [T_{C,S}]_{K_S}]_{K_{C,TGS}})$
5.	C	$\longrightarrow$	S	$([A'_C]_{K_{C,S}}, [T_{C,S}]_{K_S})$
6.	S	$\longrightarrow$	C	$([ts']_{K_{C,S}})$

図2 Kerberos システム
Fig. 2 The system of the Kerberos.

がある．また，クライアントと認証サーバ間の秘密鍵が長期にわたって使用されるため，それが悪質なクライアントの手に渡ってしまうと，その正当ユーザのチケットでサービスを受ける等の問題も考えられる．これらは共通鍵暗号を用いているために生じる問題であり，これらを公開鍵暗号を組み込むことで解決したのがYakshaである．

4. Yaksha

4.1 Yaksha システム

先に述べたKerberosには実装されていなかったデジタル署名機能だが，これを公開鍵暗号を用いることで付加したシステムがYaksha[1)]である．また，公開鍵暗号を用いるにあたって，Yakshaはシステム上に公開鍵を取り扱う機関を必要としている（**図3** 参照）．

Yakshaのシステムが果たす役割はKerberosのそれと基本的に違いはない．システム上の違いも，クライアントとサーバ間でのやりとりを共通鍵暗号の代わりに公開鍵暗号を用いて行っている点のみである．

だが，Kerberosが署名者認証が行えなかったのに対し，Yakshaではデジタル署名を利用したクライアント同士での認証や，現在注目されつつある鍵共有・鍵寄託機能をはじめとした多くの機能を多角的に拡張して運用することができる．Yakshaはこれまでに提案された機能や手続きを実装することができ，また既存のシステムに組み込むこともできる，今後のコンピュータシステムの基盤となりうるシステムであるといえる．

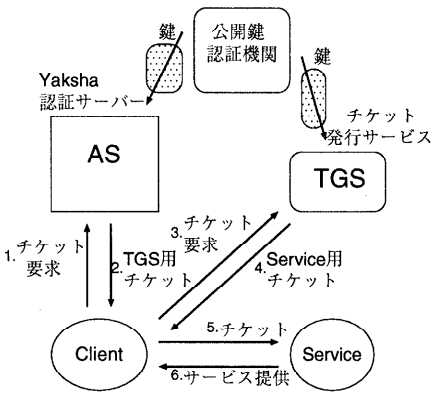


図3 Yaksha システム
Fig. 3 The system of the Yaksha.

Yakshaは公開鍵暗号（RSA方式）の秘密鍵を2つに拡張することで，サーバとクライアント間での暗号通信・署名認証を提供している．まず通常のRSA方式での暗号化・復号化操作は次のようになっている．

- 暗号化操作 $C = m^d \bmod n$
- 復号化操作 $m = c^c \bmod n$

ここで，Yakshaで拡張されたRSA方式の公開鍵と秘密鍵の関係式を示す．

$$d_1 d_2 e_1 \equiv 1 \bmod \phi(n) \qquad (2)$$

ただし，d_1，d_2 は秘密鍵，e_1 は公開鍵とする．ここで，2つの秘密鍵はそれぞれクライアント (d_1) とサーバ (d_2) が，公開鍵 (e_1) は外部の者を含む全員が知っているものとする．

このときクライアントが単独で署名を行った場合を

考えると，その署名 m^{d_1} を確認できるのは先に述べた式より d_2 と e_1 の両方を知っているサーバのみである．つまりクライアントによる署名は，サーバへの暗号文という側面を持つ．

また，サーバが単独で署名を行う場合，その署名 m^{d_2} を確認できるのは d_1 と e_1 の両方を知っているクライアントのみである．よって，この場合もクライアントへの暗号文と見ることができる．

クライアントが署名したメッセージにさらにサーバが署名した（あるいはその逆）場合には，公開鍵 (e_1, n) を用いてだれでも署名 $m^{d_1 d_2}$ を確認することができる．

Yaksha で採用している RSA 方式の変形版は，このような 3 通りの署名を生成可能である．

ここに Yaksha の一連の動作の流れを説明する．基本的な部分は Kerberos と同じであるため，C，AS，TGS，S，および K，T は前章と同じものを指すとする．ただし K はここではセッション鍵としてのみ用いられる．さらに Yaksha で用いる公開鍵のペアとして，I の持つ秘密鍵を D_I，I 以外の何者か（主にサーバ）が持つ秘密鍵を D_{Iy}，対応する公開鍵を E_I，N_I とする．Yaksha のシステムで秘密鍵を分割して所有しているのは，C と AS，AS と TGS，TGS と S である．

全体の流れも Kerberos と同じく 6 段階に分かれている．以下，Kerberos との違いを示しながら Yaksha における手続きについて説明していく．

1. C $\longrightarrow$ AS

$$(c, tgs, [[TEMP\text{-}CERT]_{D_C}, n]_{D_C})$$

Kerberos の場合と違う点は，$TEMP\text{-}CERT$ と呼ばれる使い捨ての公開鍵を用意する点である．$TEMP\text{-}CERT$ 内には (e_{temp}, n_{temp}) が格納されており，これに対応する秘密鍵 d_{temp} は C のみが知っている．これを D_C で暗号化して送る．$TEMP\text{-}CERT$ とともに入れておく n は乱数で，D_C に対する辞書攻撃を防ぐことが目的である．

2. AS $\longrightarrow$ C

$$([K_{C,TGS}]_{e_{temp}}, [T_{C,TGS}]_{D_{TGSy}}, [[TEMP\text{-}CERT]_{D_C}]_{D_{Cy}})$$

AS は D_{Cy} と E_C，N_C を知っているので，$TEMP\text{-}CERT$ を復号できる．TGS 用のチケット $T_{C,TGS}$ は AS の持つ TGS の秘密鍵 D_{TGSy} で，C と TGS との間で使用する共通鍵 $K_{C,TGS}$ は $TEMP\text{-}CERT$ 内の e_{temp} でそれぞれ暗号化する．さらに $[TEMP\text{-}ERT]_{D_C}$ に D_{Cy} で署名して，$[[TEMP\text{-}CERT]_{D_C}]_{D_{Cy}}$ を生成する．これは C や AS が間違った鍵を使用していない限り，だれにでも復号ができる．基本的には Kerberos での同手続きを公開鍵暗号を用いて行っており，相手認証は対応する鍵を用いているかどうかで判断している．

3. C $\longrightarrow$ TGS

$$([A_C]_{K_{C,TGS}}, [[T_{C,TGS}]_{D_{TGSy}}]_{d_{temp}}, [[TEMP\text{-}CERT]_{D_C}]_{D_{Cy}}, s)$$

C はまず $K_{C,TGS}$ を復号する．それを用いて，作成した認証情報 A_C を暗号化する点は Kerberos と同じである．Yaksha の場合はさらに $T_{C,TGS}$ を $TEMP\text{-}CERT$ に対応する秘密鍵 d_{temp} で署名をしておく．これは，もし仮に第 3 者が D_{TGSy} を盗み出した場合に，正しいクライアントが知らないところでチケット発行の要求をできないようにするためである．

C はこれらと，C と AS の署名つき $TEMP\text{-}CERT$，およびサービスの ID である s を TGS に送信する．

4. TGS $\longrightarrow$ C

$$([K_{C,S}]_{K_{C,TGS}}, [T_{C,S}]_{D_{Sy}}, [[T_{C,TGS}]_{D_{TGSy}}]_{D_{TGS}})$$

まず TGS は C の公開鍵を使って $TEMP\text{-}CERT$ を取り出す．さらに取り出した $TEMP\text{-}CERT$ と TGS の持つ秘密鍵 D_{TGS} を用いて送られてきたチケット $T_{C,TGS}$ を復号する．ここからの認証手順は Kerbeeros と変わらない．

C を正しいクライアントであると判断できれば，前述の Yaksha での手続き 2 と同様にしてサービス S 用のチケット $[T_{C,S}]_{D_{Sy}}$，C，S 間で用いる共通鍵 $[K_{C,S}]_{K_{C,TGS}}$ を用意する．また C が確認できるように送られてきたチケット $[T_{C,TGS}]_{D_{TGSy}}$ に秘密鍵 D_{TGS} で署名して C が復号できるようにしておく．

5. C $\longrightarrow$ S

$$([A'_C]_{K_{C,S}}, [[T_{C,S}]_{D_{Sy}}]_{e_{temp}}, [[TEMP\text{-}CERT]_{D_C}]_{D_{Cy}})$$

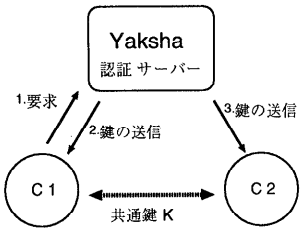
ここでは Yaksha での手続き 3 と同様に作業を行う．

6. S $\longrightarrow$ C

$$([[T_{C,S}]_{D_S}]_{D_{Sy}})$$

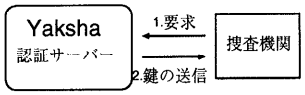
クライアント認証は手続き 4 と同様．認証ができたら，C にサービスを提供する．

また Kerberos での同手続きと同様に，クライアント C がサービス側の認証を要求した場合も考えられている．その際には認証情報 A' 内のタイムスタンプ



1. C1 ⟶ AS $c2$（相手の ID）
2. AS ⟶ C1 $[K]_{D_{Cy}E_C}$
3. AS ⟶ C2 $[K]_{D_{C2y}E_{C2}}$
4. C1 ⟶ C2 $[m]_K$

図 4 鍵共有
Fig. 4 Joint ownership of the keys.



1. A ⟶ AS $[R]_{D_A}$
2. AS ⟶ A $[K]_{D_{Ay}E_A}$

図 5 鍵供出
Fig. 5 Delivery of the keys.

の代わりに，チケット $[T_{C,S}]_{D_{Sy}}$ に D_S で署名を施したものを送信する．

4.2 Yaksha での鍵寄託方式

まず簡単に鍵共有，鍵寄託について説明しておく．鍵共有とは，鍵を保管する機関が秘密通信路を確保したい 2 者以上のクライアントに共通鍵（セッション鍵）を持たせることである．

また鍵寄託とは鍵を保管する機関に預けておくことである．これによりもしある秘密通信路に対する捜査が必要となった場合に，捜査機関が鍵を保管する機関からその通信路で用いられている共通鍵を借りることで秘密通信を盗聴することができる．強度の高い暗号を用いればより安全な秘密通信が可能であるが，逆にいえばそれが犯罪に利用された場合には捜査が難航することは間違いない．暗号が世間に急速に普及してゆきつつある現在において，これらは注目すべき機能である．

これら 2 つの機能を Yaksha は実装している．以下にこれらの機能における手続きを示す．クライアントを C，認証サーバを AS とする．

- 鍵共有（**図 4** 参照）

C1 に要求を受けた後，AS は C1，C2 に両者間で使用する共通鍵 K をそれぞれ暗号化（署名）して配送する．もし C2 がシステム内に属していないなら，D_{C2y} には 1 を代入する．4. では秘密通信が行われるが，実際には認証サーバを介して文書に署名を施したりすると考えられる．

- 鍵供出（**図 5** 参照）

A は関係する捜査機関，R は捜査機関の鍵供出要求文とする．AS は R が適切であると判断した場合に，必要とされる秘密鍵 K を A 向けに暗号化して送信する．

このように Yaksha における鍵寄託方式は Yaksha の認証サーバが非常に大きな役割を受け持っている．逆に考えると認証サーバにはクライアント間の通信の盗聴が可能なわけである．この点の改善を検討したモデルを後に示す．

5. ElGamal 方式による Yaksha のモデル

Yaksha のモデルでは公開鍵暗号に RSA 方式を採用していたが，今回は ElGamal 方式での実現を検討してみた．

前述のように ElGamal 方式では暗号化と復号化とでアルゴリズムが異なる．そのため RSA 方式とは違い，暗号化操作は公開鍵で，署名作成は秘密鍵でしか行えない，というように 2 つの鍵の役割がはっきりと分かれている．

問題となるのは，先に示したとおり通常の ElGamal 署名では署名認証を署名からメッセージを取り出すことをせずに行われるため，Yaksha のモデルで行われる「署名からメッセージ（鍵やチケットなど）を取り出す」というやりとりが実行できないという点である．今回我々はメッセージ回復型 ElGamal 署名[3]を導入することでこの点の解決を試みた．

メッセージ回復型の ElGamal 協力署名については 2 章で述べたとおりである．

署名の作成は，図 1 に従って**図 6** のように行う．

ElGamal 方式を用いる場合と RSA 方式での場合との相違点は，署名生成・認証時におけるやりとり（チケットの配送や確認，デジタル署名の生成）である．具体的に，RSA 方式では協力署名の作成に必要な両者間での通信回数が 1 回（一方が署名し送信したデー

Client		Authentication Server
$r_1 = g^{-k_1} \bmod p$		
$C = (c_1, c_2) = [M]_{yg^{-x_1}}$	$\xrightarrow{C, r_1}$	
		$M = c_2/{c_1}^{x_2} \bmod p$
		$r_{12} = M \cdot r_1 \cdot g^{-k_2} \bmod p$
		$s'_2 = k_2 - x_2 r_{12} \bmod (p-1)$
$s'_1 = k_1 - x_1 r_{12} \bmod (p-1)$	$\xleftarrow{r_{12}, s'_2}$	
$s_{12} = s'_1 + s'_2 \bmod (p-1)$		

図6 鍵寄託システムにおける協力署名プロトコル
Fig. 6 Protocol for the joint signatures in key escrow systems.

タに他方が署名することで協力署名が完成）であるのに対し，今回提案したElGamal方式では，両者間での通信が2回必要（一方がまずデータを送信し他方が署名した返信データに署名することで完成）である点である．今後は，この違いを生かした拡張も検討していく予定である．

6. Yakshaの鍵寄託方式の強化

従来のYakshaの鍵寄託方式はYakshaの認証サーバが共通鍵（セッション鍵）の生成・保管を行っていたが，これでは認証サーバが多くの情報を知りすぎることになる．今回はこの問題をセッション鍵を生成・保管する別の機関を用意することで解決を試みた．

全体的なシステム構成を図7に示す．図中の鍵生成機関がセッション鍵を生成・発行する機関であり，鍵保管機関が鍵の保管・盗聴した文書の復号を行う機関である．

本システムでは鍵保管機関は鍵の貸出しをしない．これは操作機関が無制限に盗聴を続けるのを防ぐためである．また，鍵保管機関に盗聴した文書の内容を知られるのを防ぐため，盗聴文書の復号にはブラインド復号化[12]を用いる．ブラインド復号化とはブラインド署名と同じような概念を持つ考えで，復号者に文書の内容を知られることなく復号してもらう，というものである．その一般的なブラインド復号化の流れを以下に示す．前提として復号化の依頼者Aliceが復号者Bobの暗号化鍵で暗号化された暗号文 $C_B = E_B[m]$ を保持しており，Bobに文書内容 m を知られないように C_B を復号化してもらうものとする．

(1) Alice → Bob :
Aliceは，暗号文 C_B にさらに秘密鍵（乱数）Aによる計算を施し，これを暗号文 $C_{AB} = E_A[C_B]$ とする．そして，その暗号文 C_{AB} をBobに送付する．

(2) Bob → Alice :
Bobは，送付された暗号文 C_{AB} に対して暗号

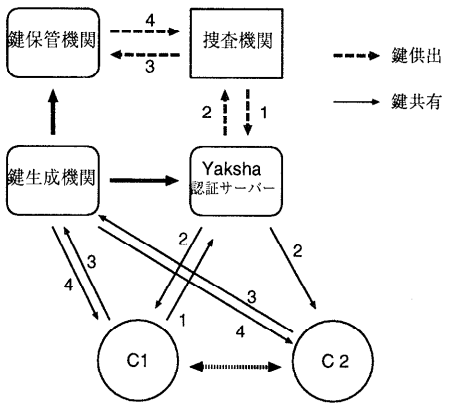


図7 改良した鍵寄託方式
Fig. 7 The modified key escrow system.

文 C_B に対する秘密鍵Bで復号化する．これを，暗号文 C_A としてAliceに返送する．ただし，ブラインド復号化は以下に見られるように，計算順序を入れ替えても結果が変わらない暗号系であることに注意する．

$$\begin{aligned} C_A &= D_B[C_{AB}] \\ &= D_B[E_A[C_B]] \\ &= D_B[E_A[E_B[m]]] \\ &= D_B[E_B[E_A[m]]] \\ &= E_A[m]. \end{aligned}$$

(3) Alice :
Aliceは，暗号文 C_A に対する秘密鍵Aで C_A を復号化して平文 m を得る．

$$\begin{aligned} D_A[C_A] &= D_A[E_A[m]] \\ &= m. \end{aligned}$$

今回考案した鍵共有・鍵供出の具体的な手続きについて説明する．まず，以下のような準備をしておく．

- 鍵生成機関はセッション鍵を作成するごとに，認証サーバにセッション鍵のIDおよび自身の秘密鍵で暗号化したセッション鍵を送信する．
- 鍵生成機関はセッション鍵を作成するごとに，鍵保管機関にその鍵と鍵のIDを送信する．

- 鍵生成機関は認証サーバ，鍵保管機関に配送済みのセッション鍵および鍵 ID を削除する．

ここで，C1 と C2 は 2 者間通信を行うクライアント，AS は認証サーバ，G は鍵生成機関，T は鍵保管機関，A は関係する捜査機関であり，セッション鍵を K，K の ID を k，鍵生成機関 G の秘密鍵を K_G，i の公開鍵を Y_i，秘密鍵を X_i，セッション鍵の復号要求文を r，捜査機関の鍵供出要求文を R とする．また盗聴した暗号文を C とし，その平文を M とする．

- 鍵共有
  1. C1 $\longrightarrow$ AS　$c2$ (C2 の ID)
  2. AS $\longrightarrow$ C1　$[[K]_{K_G}]_{X_{C1y}}, [r]_{X_{C1y}}$
     AS $\longrightarrow$ C2　$[[K]_{K_G}]_{X_{C2y}}, [r]_{X_{C2y}}$
  3. C1 $\longrightarrow$ G　$[[K]_{K_G}]_{X_{C1y}}, [r]_{X_{C1y}X_{C1}}$
     C2 $\longrightarrow$ G　$[[K]_{K_G}]_{X_{C2y}}, [r]_{X_{C2y}X_{C2}}$
  4. G $\longrightarrow$ C1　$[K]_{X_{C1y}}$
     G $\longrightarrow$ C2　$[K]_{X_{C2y}}$
- 鍵供出
  1. A $\longrightarrow$ AS　$[R]_{X_A}$
  2. AS $\longrightarrow$ A　$[k]_{X_{Ay}Y_A}$
  3. A $\longrightarrow$ T　$[R, k, [C]_B]_{Y_T}$
  4. T $\longrightarrow$ A　$[M]_B$

鍵共有については，3〜4 の手続きにおいてクライアント C1，C2 は鍵生成機関 G に鍵 K を知られることなく復号してもらうことになる（ブラインド復号化）．

また鍵の供出については，便宜上「鍵供出」と表記してあるが実際にはセッション鍵の供出は行わず，鍵保管機関 T に盗聴したメッセージをブラインド復号化してもらうにとどめている．これはクライアントの通信を無制限に盗聴されるのを防ぐためである．また手続き 3，4 に現れる B は，捜査機関 A によるブラインドファクタである．B は乱数で生成される．これを用いて暗号文 C の暗号化を行い，平文 M の内容を鍵保管機関に知られないようにする．

しかし，この方式でも認証サーバは捜査の対象となる人物については知りうるので，対象人物に盗聴捜査されていることを伝えたり，第 3 者にその人物が盗聴捜査を受けていることを知らせる可能性がある．この点の改善が今後の検討課題である．

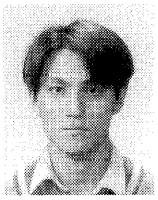
7. ま と め

メッセージ回復型協力 ElGamal 署名の安全性の検討を行っていく予定である．また，今回提案した鍵寄託方式に残された問題点の改善も検討していく．

参 考 文 献

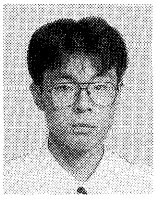
1) Ganesan, R.: Yaksha: Augmenting Kerberos with Public Key Cryptography, *Proc. ISOC Symp. on Network and Distributed System Security* (Feb. 1995).
2) Ganesan, R.: The Yaksha Security System, *Comm. ACM* (Mar. 1996).
3) Nyberg, K. and Rueppel, R.A.: Message Recovery for Signature Schemes Based on the Discrete Logarithm Probrem, *Advances in Cryptology-EUROCRYPT'94*, LNCS, Vol.950 (1994).
4) 宮地充子：強化されたメッセージ復元型署名，*Proc. SCIS96* (Feb. 1996).
5) Kohl, J. and Neuman, C.: The Kerberos Network Authentication Service, *Internet RFC 1510* (Sep. 1993).
6) Neuman, B.C. and Ts'o, T.: Kerberos: An Authentication Service for Computer Networks, *IEEE Communications* (Sep. 1994).
7) Boyd, C.: Digital Multisignatures, *Cryptography and Coding*, Clarendon Press, Oxford (1989).
8) 新保　淳：多重署名に適した ElGamal 署名の一変形方式，*Proc. SCIS94* (Feb. 1994).
9) Rivest, L.R., Shamir, A. and Adelman, L.: On Digital Signatures and Public-key Cryptograpy, *Comm. ACM* (Jul. 1978).
10) 山根義則，櫻井幸一：プライバシーを考慮した鍵寄託（Key Escrow）方式，信学技報，ISEC95-28 (1995).
11) 山根義則，櫻井幸一：無制限な盗聴を防ぐ鍵寄託方式，*Proc. SCIS96* (Feb. 1996).
12) 山根義則，古屋聡一，櫻井幸一：エルガマル暗号によるブラインド復号化方式，平成 7 年度（第 48 回連合大会）電気関係学会九州支部連合大会講演論文集 (1995).

(平成 9 年 4 月 3 日受付)
(平成 9 年 7 月 1 日採録)



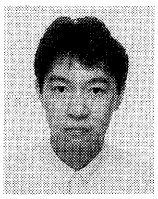
宮崎　真悟

平成 9 年九州大学工学部情報工学科卒業．現在，同大学大学院システム情報科学研究科修士課程 1 年．暗号理論，特に電子現金と鍵寄託システムの設計を研究．



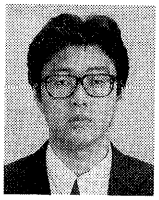
石本　関

平成9年九州大学工学部情報工学科卒業．暗号理論，情報セキュリティに関する研究に従事．現在，ブラザー工業（株）に勤務．



新保　淳（正会員）

昭和60年東京大学工学部電気電子学科卒業．昭和62年同大学大学院修士課程修了．同年（株）東芝入社．以来同社研究開発センターにて暗号，情報セキュリティの研究開発に従事．電子情報通信学会会員．



櫻井　幸一（正会員）

昭和61年九州大学理学部数学科卒業．昭和63年同大学工学研究科応用物理専攻修了．同年三菱電機（株）入社．現在，九州大学大学院システム情報科学研究科助教授．計算複雑性理論，暗号理論，情報セキュリティの研究に従事．「暗号理論の基礎」(1996年共立出版，監訳)，「数論アルゴリズムと楕円暗号理論入門」(1997年シュプリンガー東京，訳)．工学博士．電子情報通信学会，日本数学会各会員．