社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－GN－50　(1)
2004－ EIP －22　(1)
2004／1／15

個人情報の扱いを考慮したアクセス履歴証明システム

中村 めぐみ [1], 朝倉 啓充 [1], 成田 干城 [2], 重野 寛 [2], 岡田 謙一 [2]

概 要

インターネットを利用する際，個人情報漏洩などの不安から，匿名でアクセスをしたいという要求がある．一方サービス提供者は，どのようなユーザがどのように使用しているかといった情報を収集することにより，より良いサービスを提供したいという要求がある．そのため，ユーザが様々なシステムを使用すると，システムが結託することにより同一ユーザが公開した情報を収集されてしまう可能性があるので，ユーザの特徴情報と行動情報の結び付きを秘匿できるような認証方式が提案されてきた．しかし，ただ匿名性を提供するだけの個人情報保護手法では，匿名者が繰り返しアクセスをしても，それぞれが匿名であるために同一ユーザの繰り返しであることが証明できず，アクセス数によるサービスの提供などができない．そこで本稿では，匿名状態のままで匿名者同士がアクセス履歴を証明する手段を提案する．

Access History Authentication Method for Personal Data Protection

Megumi Nakamura, Hiromitsu Asakura, Tateki Narita, Hiroshi Shigeno, Ken-ichi Okada

Since there is a possibility that personal information may be revealed when using the Internet, there is demand of wanting to access using anonymity. On the other hand, those who offer service think that he wants to offer better service by investigating what user is using service how. If a user uses various systems, a user's information is collectable because a system coordinates. Therefore, the system which keeps secret connection of a user's personal information and action information was proposed. However, if anonymity is used, it cannot be shown even if a user accesses a system repeatedly. Then, in this paper, we propose the method of showing the history that anonymity persons accessed.

1　はじめに

インターネットにおいて WWW サーバへアクセスしたり，ファイル交換を行なったりする際，個人情報漏洩などの不安から，匿名でアクセスをしたいという要求がある．そのため，ユーザとしては，サービス利用時の認証行為から必要以上に個人情報を開示することを防ぐ必要があり，また，複数システムへのアクセスの結果からユーザの開示した情報の収集を不可能にするような機構が必要である．

一方，サービス提供者は，どのようなユーザがどのようにサービスを利用しているかを調査することにより，より良いサービスを提供したいという要求がある．そのため，ユーザの行動や嗜好に関する情報のプロファイリングが多々行なわれている．またユーザ管理を行なうことで，不適切な行動をとるユーザを取り締まる必要がある．

これらの要求に対して，現在普及している認証・アクセス技術を適応することは容易ではない．たとえば，ユーザ ID・パスワード方式のようなシステムでは，個人情報の登録を要求される場合がある．公開鍵証明書や属性証明書に関して

[1] 慶應義塾大学 大学院 理工学研究科
Graduate School of Science and Technology, Keio University
[2] 慶應義塾大学 理工学部
Faculty of Science and Technology, Keio University

も，これらに掲載されている個人情報が，システムのアクセス制御の実態によらず認証のたびに提示されるため，意図せぬ場面で必要以上に情報を公開してしまう可能性がある．ユーザが様々なシステムを使用すると，システムが結託することにより同一ユーザが公開した情報を収集されてしまう可能性がある．また，公開鍵認証局に登録している公開鍵を使用する場合，鍵を識別子としたユーザの追跡が有り得る．

そこで，たとえユーザの特徴情報が知られていたとしても，このユーザの行動情報との結び付きを秘匿できるような認証方式が提案されてきた．

しかし，ただ匿名性を提供するだけの個人情報保護手法には大きな欠点がある．匿名者はその匿名性ゆえに，本来自分が持っている権利から導かれる利益を享受できない可能性がある．一般に匿名者は自分の行動についての証明を行なうことができない．例えば，匿名者が繰り返しアクセスをしても，それぞれが匿名であるために同一ユーザの繰り返しであることが証明できない．そのため，アクセス回数に応じたサービスの提供などが不可能である．

そこで本稿では，匿名状態のままで匿名者同士がアクセス履歴を証明する手段を提案する．

2 プライバシ保護

2.1 関連技術

通信におけるプライバシ保護技術は，様々な立場から研究されている．

受信者に対して送信者の通信路を秘匿することを目的としたものに，中継端末で変換を行なう方式 (文献 [1]) や送信者が特定できないようなグループを用いる方式 (文献 [2]) がある．

認証においてプライバシを保護する技術には，PKI(Public Key Infrastructure) ベースによる権利認証を行なう方式 (文献 [3],[4]) や，ブラインド署名を用いる方式 (文献 [5],[6]) がある．

しかし，これらの技術ではユーザを特定できる情報を開示しないため，ユーザは行動の連続性を示すことができない．

そこで，ユーザの行動を把握する技術として cookie[7] がある．cookie は Web サーバがデータ片をクライアント端末に送信し，クライアント端末で保存する．次セッションで蓄積されたデータを利用することで，複数のセッション間でユーザに関するデータを使い回すものである．

2.2 問題点

Freenet[8][9] のような匿名 P2P システムでは，通信路は暗号化されているが，ユーザ認証を行なわないため，不正行為を取り締まることが非常に困難である．そこで，不正行為を取り締まることができるようなプライバシ保護技術が数多く研究されている．しかし，ユーザが特徴情報や識別情報を開示しないので，自己の行動を他者に示すことは不可能である．

認証や権限証明に静的な鍵を用いるシステムでは，鍵を追跡することでユーザを追跡できる．また，公開鍵証明書のように鍵に ID 情報が直接関連付けられている場合，証明書そのものからユーザの特定が可能になる．したがって，認証行為からユーザの特定や追跡を不可能とするには，アクセスのたびに鍵を更新する，鍵そのものに直接 ID 情報を関連付けない，などの対策をとる必要がある．その状態でユーザの行動を証明できれば，ユーザ主導でアクセス履歴を証明することができる．

ユーザの行動を把握するために cookie を用いる方法もあるが，ユーザが個人情報を WWW サイトに公開する場合，公開する情報の内容はサイトによって決められており，ユーザに交渉の余地はない．また，ユーザが個人情報を開示する際，それらを公開するかどうかについて，アクセスごとにユーザが確認する必要がある．また，ユーザが制御できるのは cookie の ON/OFF のみである．

本研究は，個人を特定するような情報を用いることなく，ユーザ主導でアクセス履歴を示すことができるようなシステムを提案する．

3　アクセス履歴証明システム

3.1　方針

本システムではアクセス履歴を証明するために，次のような特徴を持つアクセス履歴証明書 (以下，アクセス証と記す) を作成し，使用する．

1. 認証時の特徴情報の流出制御
   ユーザ自身が，認証時にどのような特徴情報を検証者に伝えるかを制御できる．ユーザの特徴情報や行動情報をシステムに対して必要以上に与えることを防ぐ．
2. 行動の追跡不能性とその制御
   ユーザが，システムに与える情報から行動が把握されうる範囲を制御できる．
3. アクセス証の再利用性
   アクセス証は何度でも利用できる．
4. アクセス証の利用者制限
   アクセス証はその所有者以外は利用できず，他人へ譲渡不可能である．

3.2　想定する利用形態

今後ノート PC や PDA など個人専用のモバイル端末使用が増加することを想定する．そのような環境として，頻繁にネットワーク環境や IP アドレスが変化することを想定し，環境に依存しないシステムを構成する．

利用者登録を行なってからサービスを利用するようなハイブリッド P2P 環境において，メッセージ交換やファイル交換を行なうことを考える．

応用として，P2P ネットワークにおいて匿名通信を行なっている者同士が，互いにアクセス履歴の有無を確認することで，やりとりする情報への信頼性が変化することが考えられる．また，アドホックネットワークなどを考えた場合，アクセスの有無を優先度情報としてルーティング制御へ反映させることも考えられる．

3.3　アクセス履歴証明書

匿名者同士が以前にアクセスしたことがあるかどうかを示すためのアクセス証 ($Access$) を発行する．

アクセス証には以下の要件を設定する．

- 発行者が正しく発行したことを確認できること
- 所有者しか持ち得ない情報であること
- ユーザ個人を特定する情報を含まないこと

アクセス経験の有無を公開鍵認証方式を用いて示す．

アクセス証を式 (1) のように構成する．$Pub(X)$ は X を公開鍵で暗号化・復号することを表し，$Priv(X)$ は X を秘密鍵で暗号化・復号することを表す．$H(X)$ は一方向性ハッシュ関数により X のハッシュ値を計算することを表す．

$$Access = Pub(X) + Priv(H(Pub(X))) \quad (1)$$

式 (1) はデジタル署名方式となっており，改竄の有無を検査できる．さらに秘密鍵 $Priv$ を所有している場合のみ，$Pub(X)$ から X を得ることができる．そのため，発行者が $Priv$, Pub を用いてアクセス証を構成した場合，同じ発行者のみがそのアクセス証を検証することが可能となる．

次に，所有者の認証を行なうために，式 (2) を用いる．アクセス証に含まれるメッセージを $C(Contents)$ とする．例えばアクセス回数などである．

$$X = Priv_2(H(C)) + C \quad (2)$$

ここでは鍵を所有しているかどうかに関わらず，C を得ることができる．しかし X を生成したユーザが，検証時のユーザであるかどうかを判断できないので，チャレンジ&レスポンス方式を用い，検証者が生成した $H(C)$ を提示者自身が暗号化し，$Priv_2(H(C))$ と一致するかどうかを検証する．

3.4　プロトコル

3.4.1　利用者登録

利用者登録にはブラインド署名方式を用いる．普通のデジタル署名では，署名する人が署名されるメッセージの内容を確認するが，ブラインド

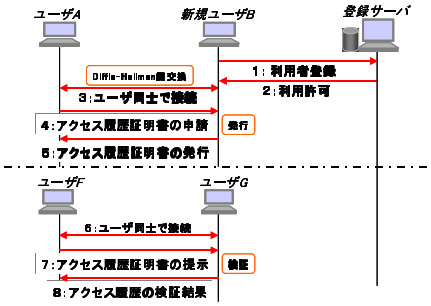

図 1: プロトコルの概要

署名では，メッセージの内容を見ることなく署名する．

例えば，名前や公開鍵，メールアドレスなどの情報をサーバに提示する．個人情報 (属性情報) である年齢や性別などの情報 (H) はサーバに公開せずに署名を受ける．するとサーバは，名前や公開鍵などの情報と，年齢や性別の情報の対応付けができないため，サービスを利用するユーザを追跡することができない．

署名を受けた個人情報はサーバの公開鍵で復号可能なので，他ユーザは個人情報 (H) を得ることができるが，その情報からサーバに登録した名前や公開鍵といった情報を入手することはできない．

ユーザ B がサーバ S に利用者登録を行なう場合について説明する．

ユーザ：ブラインド化　ユーザ B は式 (3) を用いて，署名を受けたい内容 H に対して乱数 $Rand$ を用い，ブラインド化を行なう．Z をサーバ S へ送信する．

$$Z = H \times Pub_S(Rand) \tag{3}$$

サーバ：署名　サーバ S は式 (4) を用いて，Z に対して署名を施し，Q をユーザ B に送信する．

$$Q = Priv_S(Z) \tag{4}$$

ユーザ：ブラインド解凍　ユーザ B は式 (5) の計算によってブラインド処理を解凍する．

$$\begin{aligned} & Q \times Rand' \\ = \; & (Priv_S(H \times Pub_S(Rand))) \times Rand' \\ = \; & (Priv_S(H) \times Rand) \times Rand' \\ = \; & Priv_S(H) \end{aligned} \tag{5}$$

ここで

$$Rand \times Rand' \bmod n = 1 \tag{6}$$

である．n は $Pub_S = (e, n)$ から得る．

以上の手順により，ユーザ B はサーバ S に H を見られることなく，サーバ S から署名を得て，利用証明書 $Priv_S(H)$ を受け取る．

3.4.2　サービス利用

ユーザ B はサービス利用時に利用証明書 $Priv_S(H)$ を相手に提示する．サーバ S に限らず，サーバ公開鍵 Pub_S を持っている者は誰でも検証可能である．

ここではユーザ A に提示する．ユーザ A はサーバ S の公開鍵 Pub_S を用いて復号することで，H を得る．H がサービス利用条件と一致するかどうか検証し，合致した場合はアクセスを許可する．

3.4.3　通信内容の暗号化

セッション鍵 K を Diffie-Hellman 鍵交換プロトコルで生成し，通信路を暗号化する．

3.4.4　アクセス証の発行

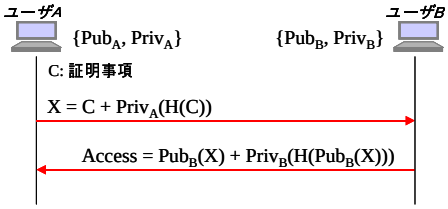

図 2: アクセス履歴証明書の発行

接続者はアクセス証 ($Access$) の発行を相手に依頼する．

ユーザ A がユーザ B に対して発行を依頼する場合について説明する．ここでユーザ A は公開

鍵ペア $(Pub_A, Priv_A)$ を，ユーザ B は $(Pub_B, Priv_B)$ を使用する．

まずユーザ A は式 (7),(8) のどちらかにより X を生成し，ユーザ B へ送信する．式 (8) の場合は $Access$ の検証時に C に含まれる変数を更新することが可能である．その必要がない場合は式 (7) を用いる．

$$X = Priv_A(H(C)) + C \tag{7}$$

$$X = Priv_A(C) + C \tag{8}$$

ユーザ B は X を用いて，式 (9) により $Access$ を生成し，ユーザ A に $Access$ を発行する．

$$Access = Pub_B(X) + Priv_B(H(Pub_B(X))) \tag{9}$$

ユーザ A は受信した $Access$ を適当な名前で保存しておく．ここでは $Access_B$ としておく．

3.4.5 アクセス証の使用

ユーザ環境の変化により，匿名ユーザ A, B が匿名ユーザ F, G へ変化したとする．

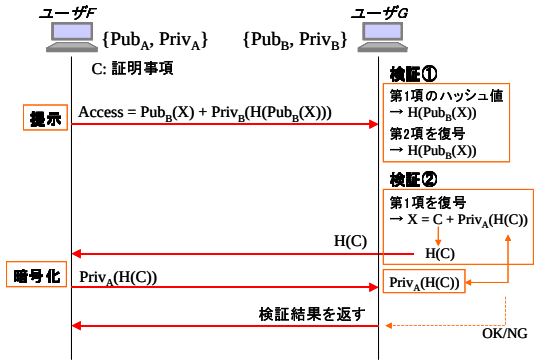

図 3: アクセス履歴証明書の検証

ユーザ F がユーザ G に対して，アクセスしたことがある相手であるかどうかを検証するため，保有する $Access$ を提示する．ユーザ F は鍵ペア $(Pub_A, Priv_A)$ を保有しているものとし，$Access_B$ を提示する場合について説明する．

ユーザ F から $Access_B$ を受信したユーザ G は，自分の保有する鍵を用いて検証を行なう．

ユーザ G が鍵ペア $(Pub_B, Priv_B)$ を保有している場合 (ユーザ B とユーザ G が同一の場合)，式 (10) により，改竄の有無を検査する．また，同時に自らの鍵で発行したものであることが確認できる．式中の?=は一致するかどうかの検証を示す．

$$\begin{aligned} \text{式 (9) 第 1 項より} \quad &= \quad H(Pub_B(X))_1 \\ \text{式 (9) 第 2 項より} \quad &= \quad Pub_B(Priv_B(H(Pub_B(X)))) \\ &= \quad H(Pub_B(X))_2 \\ H(Pub_B(X))_1 \quad ?= \quad &H(Pub_B(X))_2 \end{aligned} \tag{10}$$

次にユーザ G は，提示者 (ユーザ F) が $Access_B$ の正当な所有者であるかを確認する．式 (9) 第 1 項を復号することで X を取り出す．

X が式 (7) の場合は，C のハッシュ値 $(H(C))$ を計算してユーザ F に送信する．ユーザ F は式 (11) を用いて C' を計算し，ユーザ G に送信する．

$$C' = Priv_A(H(C)) \tag{11}$$

X が式 (8) の場合は，C をユーザ F に送信する．ユーザ F は，情報 C を更新する場合は式 (12) を用い，C' をユーザ G に送信する．

$$C' = C_{new} + Priv_A(C_{new}) + Priv_A(C) \tag{12}$$

ユーザ G は式 (13)(14) より，受信した C' が正当なものかどうか検査する．

$$\text{式 (7) 第 1 項 ?} = \ Priv_A(H(C)) \tag{13}$$

$$\text{式 (8) 第 1 項 ?} = \ Priv_A(C) \tag{14}$$

等しい場合はユーザ F が $Access_B$ の正当な所有者であることが確認できるので，ユーザ F に検証成功の通知を出す．

以上のプロセスから，以前アクセスしたことのある相手であるかどうかを判別できる．

3.5 セキュリティについての考察

3.5.1 偽造・改竄

偽造や改竄については，発行者によるデジタル署名方式を用いているので検出可能である．攻撃者が不正にアクセス証を入手しても，発行者及び申請者の公開鍵ペアを所持していなければ改竄することができない．また，不正な鍵で改竄した場合は検証時に検出することができる．

3.5.2 不正利用

アクセス証の不正利用が行われた場合は，式 (13) の認証に失敗するので検出可能である．また，通信路を Diffie-Hellman 鍵交換方式により生成した共有鍵で暗号化しているため，アクセス証を不正に入手すること自体が困難である．

3.5.3 不正なアクセス証の発行

アクセス証の生成時に申請者が不正を行なった場合，発行者が不正を行なった場合のどちらも，後に使用する段階で検出される．

3.5.4 プライバシ保護

追跡不可能性 公開鍵暗号方式を用いるが，使用する公開鍵は自身の公開鍵のみである．他人の公開鍵を使用せずに検証を行なうことができるため，公開鍵を追跡することによるユーザ追跡は有り得ない．

また，アクセス証を各ユーザに対して個別に生成するので，他のユーザと情報を共有してもユーザを特定する目印となる情報はない．

ユーザを特定する情報は使用しておらず，匿名性が保たれており，追跡は不可能であると言える．

個人情報公開制御 個人情報を扱うのは，登録サーバがユーザに対してブラインド署名によりサービス利用権利を与えるときである．このときは，ブラインド処理がされているので，登録サーバが個人情報を入手できず，不正使用することはできない．

4 実装

提案プロトコルを実際にアプリケーションと共に用いた場合，オーバヘッドなどが実用上問題にならない程度に収まることの検証を，本実装の目的とする．

リアルタイム性が求められるアプリケーションとして，チャットを実装した．ハンドルネームを使用してメッセージ交換を行なう．

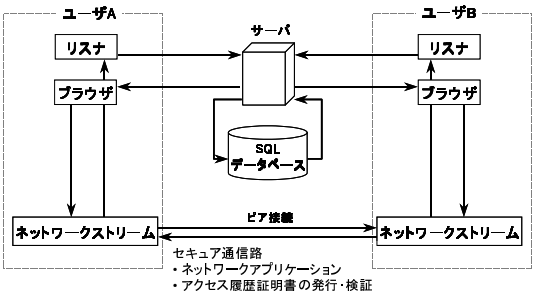

図 4: 実装システム構成

4.1 実装システム

実装システムの開発環境を表 1 に示す．

OS に依存せずに動作するプラットフォームにするため，提案システムを Java 言語を用いて実装した．

セキュリティに関して，通信路には Diffie-Hellman 鍵交換プロトコルにより共有鍵を生成した．公開鍵暗号方式として RSA を用い，一方向性ハッシュ関数として SHA-1 を用いた．

サーバにおけるユーザ登録には SQL データベースを用いた．

表 1: 実装システム技術仕様

項目	値
開発言語	Java SDK v1.4.1
Java 暗号モジュール	JCSI Provider 2.3
データベース	MySQL 4.0.15
ネットワーク環境	100Mbps の LAN プライベートネットワーク
RSA 鍵サイズ	1024bits

5 評価実験

アクセス証の発行および検証に要する時間を測定した．使用したサーバ (A) 及びクライアント (A,B) は表 2 の通りである．

アクセス証に含める情報として，100bytes, 1000bytes の 2 通りを測定し，それぞれハッシュ値を用いた場合 (式 (7)) と用いない場合 (式 (8)) の評価を行なっ

図 5: アプリケーション起動イメージ

表 2: 実験プラットフォーム

	OS	CPU	メモリ
A	linux-2.4.22	PentiumIII 1133MHz	384MB
B	Windows Me	PentiumIII 800MHz	128MB

た．以下の表中で，100bytes の情報とそのハッシュ値 (20bytes) を用いる場合を 20(100)[bytes] と表記する．

5.1 生成

アクセス証の生成に要する時間を表 3 に示す．これは表 2 におけるプラットフォーム A における値である．表中の処理内容は次の通りである．

- 申請準備: 式 (7)，式 (8) による X の生成
- 生成完了: 式 (9) によるアクセス証の生成

ネットワークの送受信処理も含め，アクセス証を申請してから入手するまでに要する時間は約 300msec であった．

表 3: アクセス証発行時間 (msec) [A の場合]

	申請準備	生成完了
20(100)	91.06	134.17
20(1000)	92.47	134.87
100	93.48	145.29
1000	94.00	132.61

表 4: 検証過程の平均処理時間 (msec) [A の場合]

	サイズ (byte)	改竄検出	暗号化	検証
失敗 1	20(100)	48.57		
	20(1000)	48.33		
	100	48.80		
	1000	49.00		
失敗 2	20(100)	45.82	71.84	0.10
	20(1000)	45.88	72.14	0.45
	100	45.36	73.30	0.06
	1000	45.91	245.45	0.19
成功	20(100)	46.84	72.46	0.14
	20(1000)	46.30	73.14	0.14
	100	46.06	73.39	0.18
	1000	45.66	246.10	0.22

5.2 検証

検証に要する処理時間について，以下の 3 パターンを測定したものを表 4,5 に示す．

1. 検証失敗 1
   発行者ではない相手に提示した場合・改竄を検知した場合
2. 検証失敗 2
   提示者が本来の所有者ではない場合
3. 検証成功

表 4 において，改竄検出は式 (10)，暗号化は式 (11)，検証は式 (13) の処理を示す．検証トランザクションの中でプラットフォーム A が処理を行なったときの値である．

表 5 は，プラットフォーム A-B 間で各処理を行ない，A もしくは B がアクセス証を提示してから検証結果を得るまでの処理時間を示す．

5.3 考察

表 3 から，アクセス証の発行に要する時間は，含まれる情報サイズに依らずほぼ一定の値を取ることがわかる．一回の接続で最大 1 つのアクセス証を発行するということを考えると，生成完了

表 5: 検証に要する平均処理時間 (msec)

		20(100)	20(1000)	100	1000
失敗 1	A	96.78	98.04	107.10	92.23
	B	74.08	72.86	73.67	77.08
失敗 2	A	286.32	560.02	288.92	741.53
	B	272.20	464.60	273.33	713.46
成功	A	305.42	565.52	309.33	757.62
	B	285.80	487.40	290.59	723.20

まで約 130msec，入手するまで約 300msec という値は実用の範囲であると考えられる．

表 4 のアクセス証の検証に要する処理時間から，失敗 1 のパターンではどのサイズでも約 50msec で検証が済んでいる．表 5 より，送受信を考慮しても 100msec 程度である．この失敗 1 のパターンは，相手がわからない場合に手持ちのアクセス証を全て提示することを考えると，最も頻度が高いと思われる．

多数のユーザと接続経験があるユーザがアクセス証を複数同時に提示した場合，アクセス証の数が 100 程度になることが考えられる．表 5 と同様の条件で 1000bytes の情報を使用した場合，プラットフォーム A で 100 個の証明書を提示し，検証した場合は 11.90sec, プラットフォーム B では 9.34sec であった．

表 5 より，処理には約 300〜750msec を要する．検証者 (発行者) が正しい場合はチャレンジ＆レスポンスを行なう．この場合は提示者 (申請者) における暗号化処理で，情報のサイズに応じて時間がかかる．これは公開鍵暗号の処理速度が遅いためである．最終的に検証に失敗しても成功しても，処理に要する時間に大きな変化はない．

これらの値は，ファイル交換やメッセージ交換などのアプリケーションを使用している時間と比較すると，短時間であると言える．

6 おわりに

本稿では，匿名者同士が匿名状態のままで互いにアクセスしたという履歴を示す方法を提案した．また，プロトタイプシステムを実装し，各処理に要する時間を測定した．その結果，検証プロセスが終了するまでの時間は約 500msec であり，また，たとえ 100 ユーザとの履歴を検証したとしても，10sec 程度である．これらの値は，アプリケーション使用時間を考えると実用範囲内であると言える．

参考文献

[1] P.F.S.amd D.M.Goldschlag and M.G.Reed; Anonymous connections and onion routing. In Proc. IEEE Symposium on Security and Privacy, Oakland, California, 1997.

[2] M.K.Reiter and A.D.Rubin; Crowds: anonymity for web transactions. ACM Transactions on Information and System Security, 1(1):66-92, 1998.

[3] 梅澤健太郎，斎藤孝道，奥乃博; プライバシーを重視したアクセス制御機構の提案, 情報処理学会論文誌，Vol.42，No.8，pp.2067-2076，Aug.2001.

[4] 梅澤健太郎，斎藤孝道，奥乃博; 権限証明書と SSL 相互認証による匿名アクセス制御方式, 情報処理学会論文誌，Vol.43，No.8，pp.2562-2572，Aug.2002

[5] 佐藤直之，鈴木英明; 耐タンパ個人端末を利用し個人情報の保護を可能とした認証方式, 情報処理学会論文誌，Vol.41，No.8，pp.2129-2137，Aug.2000.

[6] 飯田恭弘, 上野正巳, 阿川雄資; ”匿名認証方式 ”の運用に関する一検討, 情報処理学会研究報告 DPS111-26，CSEC20-26，2003.2.28.

[7] Netscape Communications. http://www.netscape.com/

[8] Freenet. http://www.freenetproject.org/

[9] I.Clarke, O.Sandberg, B.Wiley, and T.W.Hong; Freenet: A distributed anonymous information storage and retrieval system. In Lecture Notes in Computer Science, volume 2009, 2001.