Computer Security Symposium 2019
21 - 24 October 2019

アキュームレータを用いたポリシーベースのブラックリスト型匿名認証システム

新居 稜介[1,a)]　中西 透[1,b)]

概要：第三者機関を必要としない匿名認証方式としてアキュームレータを用いた評価値ベースのブラックリスト型匿名認証システムが提案されている．この方式はアキュームレータにより効率化されており，評価値に基づいてユーザの失効を行える．しかし，複数のカテゴリで評価するポリシーベースの認証に対応していないという問題がある．そこで本研究では，ポリシーを導入するとともに，カテゴリ数に依存して認証時間が増加するのを抑える効率の良い方式を提案する．さらに PC 上に実装し，本方式の有用性について示す．

キーワード：匿名認証，アキュームレータ，ペアリング

An Efficient Blacklistable Anonymous Credential System with Policy-Based Reputation Using Accumulator

Ryosuke Nii[1,a)]　Toru Nakanishi[1,b)]

Abstract: For the privacy-enhancing authentication without any trusted third party, a blacklistable anonymous credential system with reputation using an accumulator has been proposed. This system becomes efficient due to an accumulator, and users can be revoked based on the reputation. However, this system does not support the policy-based authentication using evaluates in multiple categories. In this paper, we propose an efficient system where the increase of the authentication time depending on the number of categories is suppressed. In addition, the practicality is shown by experiments based on the implementation on a PC.

Keywords: Anonymous Credential, Accumulator, Pairing

1. はじめに

1.1 背景

現在広く利用されているユーザ ID とパスワードなどの認証情報を使った ID ベース認証では，サービス提供者 (SP:Service Provider) が ID から特定のユーザのサービス利用履歴を追跡できてしまうというプライバシー問題がある．そこで ID を SP に与えずに認証することができる匿名認証が提案されている．匿名認証では，ID を管理する第三者機関 (TTP:Trusted Third Party) が正当なユーザであることを示して証明書を発行し，その証明書を用いて ID なしで認証する．ユーザの不正行為が発生した場合，SP が TTP と連携することにより不正なユーザの ID を特定する．しかし，このことは TTP と SP が結託すれば任意のユーザを追跡できてしまうということを意味し，完全なプライバシーを提供できていない．

1.2 先行研究

そこで，TPP を必要としない匿名認証方式として，ブラックリスト型匿名認証システム (BLAC)[1] が提案されている．この方式では，毎回の認証のセッションでユーザ

[1] 広島大学
Hiroshima University
[a)] m191106@hiroshima-u.ac.jp
[b)] t-nakanishi@hiroshima-u.ac.jp

自身の秘密鍵からチケット t を生成し SP に送付する．あるセッションにおいてユーザが不正を行なった場合，そのセッションの t がブラックリスト BL に登録される．さらにユーザは認証において，自分の秘密鍵に対応した t が BL に登録されていないことを証明することにより，BL に登録されているかどうかチェックでき，不正者を排除できる．しかし，BLAC におけるこの証明はブラックリストに登録されているすべてのチケットに対して行うため，計算時間が BL のサイズに比例してしまい非効率である．また，ユーザの不正行為の程度に依らず，一度の不正でブラックリスト化されてしまう．

そこで BLAC から発展した研究として，[2] では BLAC の計算時間の削減が行われている．この方式ではユーザの過去のセッション ID リストと BL に登録されているセッション ID リストのアキュームレータに圧縮することにより，一つの検証式で認証を可能としている．また，[3] では不正行為の種類ごとに評価値を設定し，不正行為の程度を評価値の和で評価できるように拡張が行なわれている．さらに，複数のカテゴリに分けてそれぞれのカテゴリで評価を行う認証に対応しており，認証時には各カテゴリで決められたポリシーを評価値の和が満たすか否か，さらに各ポリシーで構成される論理式を満たすか否かで認証を行うことができる．[4], [5] では [2] のアキュームレータベース方式を評価値ベースに拡張している．この方式では，ユーザの積算された評価値がある範囲にあることをアキュームレータと範囲証明により効率的に行なうことができる．

1.3 本研究について

本研究では，アキュームレータベース方式 [4], [5] で対応できていなかった複数カテゴリに分けて各カテゴリのポリシーに基づいて認証を行う方式への拡張を行う．ただし，ポリシーの論理式として AND のみを考える．単純な方法として，各カテゴリで [5] の方式の処理を行いポリシーの論理式を構成することが考えられるが，処理時間がカテゴリ数に依存してしまう．そこで各カテゴリの評価値の和をビット結合して一つの値とし，その値について範囲証明を行うことによりポリシーの論理式を構成することを考える．この方法での処理はアキュームレータの検証と範囲証明が一回となるため処理時間がカテゴリ数に依存しなくなる．しかし範囲証明で事前に準備する署名の総数がカテゴリ数のべき乗に依存してしまう．この署名は全て SP とユーザが保持しなければならないため，この署名のデータサイズが問題となる．

そこでこの二つの手法を基に，処理時間と署名のデータサイズについてバランスをとった手法を考える．そのために，全カテゴリを任意の数ごとにブロック化する．そして，ブロックごとに評価値の和をビット結合して範囲証明を行い，ポリシーの論理式を構成する手法を提案する．さらに提案方式を PC 上で実装し，処理時間及び署名のデータサイズを測定し，実用的なパラメータ設定の検討とともに評価を行う．

2. 数学的準備

2.1 双線形写像

本研究では以下の双線形群を利用する．$\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ を位数 p の巡回群とする．$\mathbb{G}_1, \mathbb{G}_2$ の生成元をそれぞれ g, h とする．このとき双線形写像 e: $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ は以下の双線形性と非退化性を満たす．

- **双線形性**: 任意の $P, Q \in \mathbb{G}_1, R, S \in \mathbb{G}_2$ について
  $e(PQ, R) = e(P, R)e(Q, R)$
  $e(P, RS) = e(P, R)e(P, S)$
- **非退化性**: $e(g, h) \neq 1$

上記の双線形写像は楕円曲線上のペアリングにより実現できる．

2.2 知識の署名

本研究では，秘密情報の証明に知識の署名 (SPK:Signature based on Proof of Knowledge) を用いる。SPK は知識の証明 (PK:Proof of Knowledge) をハッシュ関数により非対話型に変換することで得られる．

知識のゼロ知識証明とは証明者 P と検証者 V の対話型プロトコルで，ある関係を満たす秘密情報を知っていることを秘密情報を漏らすことなく証明する．離散対数の秘密情報 x を知ることを示すメッセージ m における SPK は以下のように記述される．

$$SPK\{(x) : y = g^x\}(m)$$

2.3 AHO 署名

AHO 署名は複数の群要素のメッセージに署名できる方式である．また署名検証のペアリングの関係式をゼロ知識証明できる．本研究ではメッセージの個数を 4 つの場合と 1 つの場合としている．

この AHO 署名は下記の 3 つのアルゴリズムで構成される．以下はメッセージが 4 つの場合を示す．1 つの場合も同様である．

(1) **AHOKeyGen**: AHO 署名の秘密鍵を出力する．素数の位数 p である双線形群 $\mathbb{G}$ を選び，その双線形写像を e とする．$h, G_r, H_r \in \mathbb{G}, \mu_z, \nu_z, \mu_1, \mu_2, \mu_3, \mu_4, \nu_1, \nu_2, \nu_3, \nu_4, \alpha_a, \alpha_b \in Z_p$
をランダムに選ぶ．これらを使って以下を計算する．
$G_z = G_r^{\mu_z}, H_z = H_r^{\nu_z}, G_1 = G_r^{\mu_1}, ..., G_4 = G_r^{\mu_4}, H_1 = H_r^{\nu_1}, ..., H_4 = H_r^{\nu_4}, A = e(G_r, h^{\alpha_a}), B = e(H_r, h^{\alpha_b})$
出力:
AHO 署名の公開鍵:
$pk_{AHO_4} = (h, G_r, H_r, G_z, H_z, G_1, ..., G_4, H_1, ..., H_4)$
AHO 署名の秘密鍵:

$sk_{AHO_4} = (\alpha_a, \alpha_b, \mu_z, \nu_z, \mu_1, ..., \mu_4, \nu_1, ..., \nu_4)$

(2) **AHOSign**: 4 つのメッセージ (M_1, M_2, M_3, M_4) と，上記の秘密鍵 sk_{AHO} から署名を作成する．まず $\beta, \epsilon, \eta, \iota, \kappa \in Z_p$ をランダムに選び，以下のように $\theta_1, ..., \theta_7$ を計算する．

$\theta_1 = h^{\beta}, \theta_2 = h^{\epsilon - \mu_z \beta} \prod_{i=1}^{4} M_i^{-\mu_i}, \theta_3 = h^{\eta}, \theta_4 = h^{(\alpha_a - \epsilon)/\eta}, \theta_5 = h^{\iota - \nu_z \beta} \prod_{i=1}^{4} M_i^{-\nu_i}$,

$\theta_6 = h^{\kappa}, \theta_7 = h^{(\alpha_b - \iota)/\kappa}$

出力:署名 $\sigma = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)$

(3) **AHOVerify**: 上記のメッセージ (M_1, M_2, M_3, M_4) と署名 σ に対し検証を行う．以下の検証式を満たせば正当な署名として受理する．

$A = e(G_z, \theta_1) e(G_r, \theta_2) e(\theta_3, \theta_4) \prod_{i=1}^{4} e(G_i, M_i)$

$B = e(H_z, \theta_1) e(H_r, \theta_5) e(\theta_6, \theta_7) \prod_{i=1}^{4} e(H_i, M_i)$

2.4 評価値ベースアキュームレータ

アキュームレータとは，ある集合を一つのデータに圧縮し，集合の関係を検証できる暗号プリミティブである．[2] では，2 つの集合の包含関係を検証できるアキュームレータをブラックリスト型匿名認証システムに用いることで認証時間の削減を行なっている．[2] におけるアキュームレータでは，2 つの集合の共通要素の個数が検証できる．[4], [5] では，これを評価値ベースのアキュームレータに拡張しており，共通要素に対する評価値の和が検証できる．この評価値ベースアキュームレータのアルゴリズムを以下に示す．

(1) **AccSetup**: 素数の位数 p である双線形群 $\mathbb{G}_1, \mathbb{G}_2$ を選び, その双線形写像を e とする．$g \in \mathbb{G}_1, h \in \mathbb{G}_2, \gamma \in Z_p$ をランダムに選び以下の計算を行う．

$g_1 = g^{\gamma^1}, ..., g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, ..., g_{2n} = g^{\gamma^{2n}}, h_1 = h^{\gamma^1}, ..., h_n = h^{\gamma^n}, h_{n+2} = h^{\gamma^{n+2}}, ..., h_{2n} = h^{\gamma^{2n}}$

$(p, G_1, G_2, e, g_1, ..., g_n, g_{n+2}, ..., g_{2n}, h_1, ..., h_n, h_{n+2}, ..., h_{2n})$ を公開情報とする．

(2) **AccGen**: **AccSetup** で計算された公開情報を用いてアキュームレータを計算する．集合 $\{1, ..., n\}$ の 2 つの部分集合 U, V を考える．V のアキュームレータは $acc_V = \prod_{i \in V} g_{n+1-i}^{s_i} (s_i \in Z)$ と計算される．ここで s_i とは要素 i に対応する評価値である．

(3) **AccWitGen**: アキュームレータの検証に必要な補助情報 W を計算する．

$W = \prod_{i \in U} \prod_{j \in V}^{i \neq j} h_{n+1-j+i}^{s_j}$ と計算される．

(4) **AccVerify**: **AccGen** と **AccWitGen** で計算された acc_V, W と公開情報を用いてアキュームレータを検証する．公開情報から U のアキュームレータ $acc_U = \prod_{i \in U} h_i$ を計算する．そして，acc_V, acc_U, W を用いて以下の式を検証する．

$$\frac{e(acc_V, acc_U)}{e(g, W)} = e(g_1, h_n)^{\sum_{i \in U} s_i} \tag{2.1}$$

2 つの集合 U, V はそれぞれあるユーザが持つセッション ID の集合，全ユーザのセッション ID の集合を対応させており，これによってあるユーザの評価値の和 $\sum_{i \in U} s_i$ が検証できる．

3. 従来方式

[2] では，BLAC にペアリングベースのアキュームレータを用いることにより処理の効率化が行われている．この方式では，ユーザは自分のセッションの ID のリスト L_U を持ち，SP はセッションの ID のブラックリスト L_S を持つ．そして，アキュームレータを用いて acc_{L_U}, acc_{L_S} という一つの値にまとめることにより，以下の検証式により検証される．

$$\frac{e(acc_{L_U}, acc_{L_s})}{e(g, W)} = e(g_1, g_n)^{|L_U \cap L_S|} \tag{3.1}$$

右辺の指数 $|L_U \cap L_s|$ が 0 のとき，ユーザのセッション ID はブラックリストに含まれていないことを表す．また，セッション ID のリストサイズに依らず一つの検証式の処理で認証が可能となっている．

また [3] では，BLAC の評価方法の拡張を行なっている．具体的には，セッションにおけるユーザのふるまいの良し悪しを他ユーザやサーバが評価点 s_j として評価し，これまでに積算された評価点をもとに認証を行う．またさらに，評価尺度を複数のカテゴリに分けてそれぞれのカテゴリで評価を行う認証に対応しており，認証時には各カテゴリ c_i ごとに決められたポリシー $\mathcal{P}_{ki}$ が設定される．各ポリシー $\mathcal{P}_{ki}$ にはしきい値が設定されており，そのしきい値より評価値が大きいもしくは小さいことが検証される．そして，さらに各ポリシー上の論理式 $\bigvee_{k=1}^{a} (\bigwedge_{i=1}^{m} \mathcal{P}_{ki})$ を満たすか否かの検証を行うことができる．

[4], [5] では，[2] を基に評価値ベースの方式に拡張している．そのために 2.4 節で紹介した評価値ベースアキュームレータを用いている．[4] の方式では，ユーザは SP に対して自分の評価値を公開して認証を行なっていた．この場合，その評価値の合計からユーザの特定につながる恐れがあり匿名性が弱い．

そこで [5] の方式では，SP が設定した認証される範囲にユーザの評価値が入っているかを証明する範囲証明で認証を行うように拡張されている．範囲証明のプロトコルとしては，まず SP が範囲内に含まれる全ての値について，それぞれをメッセージとして AHO 署名により署名したものをあらかじめ公開しておく．そしてユーザは，自分の評価値の和に対応する署名を選びゼロ知識証明を行う．これにより，ユーザは SP に自分の評価値の和の具体値を知られることなく，認証される範囲に入っていることを証明できる．この範囲証明により評価値の合計をもらさないことか

ら匿名性の強化がなされている．このとき，範囲証明用の署名はユーザが取りうる評価値の和の個数分必要になることが問題点としてある．

[5] の方式では，複数カテゴリに分けて各カテゴリのポリシーに基づいて認証することが考慮されていない．本研究では，[5] の方式を複数カテゴリに基づいた認証に拡張する．

4. 提案方式

4.1 アイデア

本研究では，ポリシーで用いる論理式も AND のみとする．まず単純にポリシーを導入する場合を考える．具体的には，各カテゴリごとに従来方式 [5] のアキュームレータの検証を行う．そして範囲に入っていることを範囲証明により行う．このとき，アキュームレータの検証式及び範囲証明の SPK はカテゴリ数に比例してしまう．そこで，(2.1) 式の右辺の指数である各カテゴリの評価値の和をビット結合することにより，一つの値で全カテゴリの評価値の和を表現することを考える．これにより，アキュームレータの検証式も一つとなるため，カテゴリ数に依らずに検証回数が一回となる．このように改良を行なった評価値ベースアキュームレータを次節で示す．

しかしこの改良により，一つの AHO 署名で全てのカテゴリの範囲証明を行う必要がある．そのため SP が作成する AHO 署名の総数は，各カテゴリ間でのユーザが取りうる評価値の和の組み合わせとなり，m をカテゴリ数，$\mathcal{R}$ を各カテゴリでの取りうる評価値の和の個数とすると，$\mathcal{R}\times m$ から $\mathcal{R}^m$ に増加する．ユーザと SP は認証の処理において全ての AHO 署名を保持しなくてはならないため，カテゴリ数が増えた場合，管理するデータのサイズが膨大なものとなる．この問題の解決案として，ブロック化により検証コストと署名のデータサイズのバランスをとる手法を 4.3 節で述べる．

4.2 評価値ベースアキュームレータの改良

検証に必要な各カテゴリの値をビット結合して一つの値にまとめることにより，検証回数を一回のみとしている．具体的には，セッション j のカテゴリ i の評価値 $s_{i,j}$ に対して，$S_i=\sum_{j\in V} s_{i,j}\cdot\ell_{i-1}(\ell_{i-1}=2^{z\cdot i-1}, z:$ ビットシフトする桁数) としてビットシフトし，$S=\sum_{i=1}^{m} S_i$ としてビット結合する．ビット結合することで一つの値である S から各カテゴリの評価値の和を一括して得ることができる．

このビット結合により，2.4 節のアルゴリズムにおける **AccGen**,**AccWitGen** に対して一括して計算できるようにする．またそれに伴い **AccVerify** の変更も行う．以下に改良後のアルゴリズムを示す．

(1) **AccSetup***: **AccSetup** と同じ．

(2) **AccGen***: **AccSetup***で計算された公開情報を用いてアキュームレータを計算する．集合 $\{1,...,n\}$ の 2 つの部分集合 U,V を考える．V のアキュームレータは $acc_V=\prod_{i=1}^{m}\prod_{j\in V} g_{n+1-j}^{s_{i,j}\ell_{i-1}}$ と計算される．

(3) **AccWitGen***: $S=\sum_{i=1}^{m}\sum_{k\in U} s_{i,k}\cdot\ell_{i-1}$ の検証に必要な補助情報 W は，
$W=\prod_{i=1}^{m}\prod_{k\in U}\prod_{j\in V}^{j\neq k} h_{n+1-j+k}^{s_{i,j}\cdot\ell_{i-1}}$ として計算される．

(4) **AccVerify***: **AccGen***と **AccWitGen***で計算された acc_V, W と公開情報を用いて $S=\sum_{i=1}^{m}\sum_{k\in U} s_{i,k}\cdot\ell_{i-1}$ を検証する．公開情報から U のアキュームレータ $acc_U=\prod_{k\in U} h_k$ を計算する．acc_V, acc_U, W を用いて以下の式を検証する．

$$\frac{e(acc_V, acc_U)}{e(g,W)}=e(g_1,h_n)^S$$

acc_V, acc_U, W を代入すると，

$$\begin{aligned}
&\frac{e(\prod_{i=1}^{m}\prod_{j\in V} g_{n+1-j}^{s_{i,j}\cdot\ell_{i-1}}, \prod_{k\in U} h_k)}{e(g, \prod_{i=1}^{m}\prod_{k\in U}\prod_{j\in V}^{j\neq k} h_{n+1-j+k}^{s_{i,j}\cdot\ell_{i-1}})}\\
&=\frac{e(g,h)^{\sum_{i=1}^{m}\sum_{k\in U}\sum_{j\in V}\gamma^{n+1-j+k}\cdot\sum_{i=1}^{m}\sum_{k\in U}\sum_{j\in V} s_{i,j}\cdot\ell_{i-1}}}{e(g,h)^{\sum_{i=1}^{m}\sum_{k\in U}\sum_{j\in V}^{j\neq k}\gamma^{n+1-j+k}\cdot\sum_{i=1}^{m}\sum_{k\in U}\sum_{j\in V}^{j\neq k} s_{i,j}\cdot\ell_{i-1}}}\\
&=e(g,h)^{\gamma^{n+1}\cdot S}\\
&=e(g_1,h_n)^S
\end{aligned}$$

となり，等式が成り立つ．

4.3 ブロック化による検証コストと署名のデータサイズのバランス化

本節では検証コストと範囲証明の署名のデータサイズのバランスをとる手法を提案する．このために，全カテゴリ分の評価値を結合するのではなく，全カテゴリを ϕ 個ごとにブロック化して各ブロックごとに結合する．そしてブロックごとにアキュームレータの検証及び範囲証明の SPK を行う．このときブロック a $(1\le a\le\psi)$ のアキュームレータは

$$acc_{V,a}=\prod_{i=1}^{\phi}\prod_{j\in V} g_{n+1-j}^{s_{i+(a-1)\cdot\phi,j}\cdot\ell_{i-1}} \tag{4.1}$$

となる．また補助情報は

$$W_a=\prod_{i=1}^{m}\prod_{k\in U}\prod_{j\in V}^{j\neq k} h_{n+1-j+k}^{s_{i+(a-1)\cdot\phi,j}\cdot\ell_{i-1}}$$

となる．そしてブロック a のビット結合した評価値

$$S_a=\sum_{i=1}^{\phi}\sum_{k\in U}\sum_{j\in V}^{j\neq k} s_{i+(a-1)\cdot\phi,j}\cdot\ell_{i-1}$$

に対する検証式は以下のように表される．

$$\bigwedge_{a=1}^{\psi}\frac{e(acc_{V,a}, acc_U)}{e(g,W_a)}=e(g_1,h_n)^{S_a} \tag{4.2}$$

これにより，署名の総数は $\mathcal{R}^m$ から $\psi \times \mathcal{R}^\phi$ 程度に減少する．そのため，ϕ の値を調整することにより計算時間と署名のデータサイズのバランスを検討できる．

4.4 モデル

提案方式は以下のアルゴリズム，プロトコルから成る．

- **セットアップ**: SP がアキュームレータの公開パラメータ，AHO 署名の公開鍵，秘密鍵を生成する．
- **登録**: SP はユーザが選んだ秘密鍵に対する初期証明書を発行しユーザに送付する．
- **認証**: SP がユーザにセッション ID のリスト L_S と各セッション $j \in L_S$ の各カテゴリごとの評価値 $s_{i,j}$ を送付する．ポリシーは $\bigwedge_{i=1}^{m} \mathcal{P}_i$ として表される．ここで，$\mathcal{P}_i$ はカテゴリ i のユーザの評価値の和 S_i が正当な範囲に含まれていることを表す．ユーザはこれらと証明書を用いて SP と匿名認証を行う．匿名認証が成功した場合，SP はユーザへ今回のセッション ID を割り当てるとともに，それを含めたユーザ ID 集合の証明書を発行する．

4.5 アルゴリズム

- **セットアップ**: SP は以下の処理により公開鍵と秘密鍵を生成する。
  (1) SP: **AccSetup***より，アキュームレータの公開鍵 $pk_{acc} = (g_1 = g^{\gamma^1}, ..., g_n = g^{\gamma^n}, g_{n+2} = g^{\gamma^{n+2}}, ..., g_{2n} = g^{\gamma^{2n}}, h_1 = h^{\gamma^1}, ..., h_n = h^{\gamma^n}, h_{n+2} = h^{\gamma^{n+2}}, ..., h_{2n} = h^{\gamma^{2n}})$ を生成する．
  (2) SP: AHO 署名の公開鍵と秘密鍵を計算する．**AHOKeyGen** より，以下を計算する．
  $G_z = G_r^{\mu_z}, H_z = H_r^{\nu_z}, G_1 = G_r^{\mu_1}, ..., G_4 = G_r^{\mu_4}, H_1 = H_r^{\nu_1}, ..., H_4 = H_r^{\nu_4}, A = e(G_r, h^{\alpha_a}), B = e(H_r, h^{\alpha_b})$
  AHO 署名の公開鍵は
  $pk_{AHO} = (G_r, H_r, G_z, H_z, G_1, ..., G_4, H_1, ..., H_4)$,
  AHO 署名の秘密鍵は
  $sk_{AHO} = (\alpha_a, \alpha_b, \mu_z, \nu_z, \mu_1, ..., \mu_4, \nu_1, ..., \nu_4)$ となる．
  (3) SP: 評価値の範囲に関する AHO 署名の公開鍵と秘密鍵を計算する．メッセージが 1 つの場合の **AHOKeyGen** より，以下を計算する．
  $\hat{G}_z = \hat{G}_r^{\mu'_z}, \hat{H}_z = \hat{H}_r^{\nu'_z}, \hat{G}_1 = \hat{G}_r^{\mu'_1}, \hat{H}_1 = \hat{H}_r^{\nu'_1}, \hat{A} = e(\hat{G}_r, h^{\alpha'_a}), \hat{B} = e(\hat{H}_r, h^{\alpha'_b})$
  AHO 署名の公開鍵は
  $pk'_{AHO} = (\hat{G}_r, \hat{H}_r, \hat{G}_z, \hat{H}_z, \hat{G}_1, \hat{H}_1)$,
  AHO 署名の秘密鍵は
  $sk'_{AHO} = (\alpha'_a, \alpha'_b, \mu'_z, \nu'_z, \mu'_1, \nu'_1)$ となる．
  (4) SP: パラメータ ϕ, ψ を設定する．また，ビットシフトする桁数 ℓ_k を設定する．さらに，ポリシーを $\bigwedge_{i=1}^{m} \mathcal{P}_i$ と定義して公開する．
  (5) SP: 範囲証明用の署名を計算する．カテゴリ i の評価値の和の有効な値の集合を Ω_i とする．各ブロック a に対して，$P_a = \{\sum_{k=1}^{\phi} \rho_{k+(a-1)\cdot\phi} \cdot \ell_{k-1} \mid \rho_{k+(a-1)\cdot\phi} \in \Omega_{k+(a-1)\cdot\phi}\}$ とする．sk'_{AHO} を用いて各 P_a の各 v に対して AHO 署名 $\sigma'_{a,v}$ を計算する．これらの署名を
  $\sigma'_{a,v} = (\Theta_{a,v,1}, ..., \Theta_{a,v,7})$
  とする．カテゴリ i の評価値の和の取りうる個数を n_i とするとき，署名の総数は $\mathcal{S} = (\prod_{i=1}^{\phi} n_i) + (\prod_{i=\phi+1}^{2\phi} n_i) + \cdots + (\prod_{i=(\psi-1)\phi+1}^{m} n_i)$ 個となる．
  (6) SP: 各ユーザのセッション ID の集合を L_U, 全ユーザのカテゴリ i の使用済みセッション ID の集合を L_S とする．また，カテゴリ i のセッション ID j での評価値を $s_{i,j}$ とする．セットアップの段階では各 $L_S = \emptyset$ とする．
- **登録**: ユーザは以下のプロトコルにより SP から初期証明書とその秘密鍵を取得する．
  (1) ユーザ: 各ユーザは以下のパラメータをランダムに選ぶ．
    · ユーザの秘密情報 x
    · 証明書のタグ T_0
    · $r_{x,0}, r_{T_0} \in Z_p^*$($x$ と T_0 の SPK に使用．)
  (2) ユーザ: ユーザはコミットメントとして $C_{x,0} = h^x \hat{h}^{r_{x,0}}, C_{T_0} = h^{T_0} \hat{h}^{r_{T_0}}$ を計算し，SP へ $(C_{x,0}, C_{T_0})$ を送付．また，$SPK\{(x, T_0, r_{x,0}, r_{T_0}) : C_{x,0} = h^x \hat{h}^r_{x,0}, C_{T_0} = h^{T_0} \hat{h}^r_{T_0}\}$ を計算し送付する．
  (3) SP: ユーザの使用済みセッション ID の集合を $L_U = \emptyset$ とするまた $P_0 = h_1, R_0 = 0, r_{R_0} = 0$ とする．コミットメントとして $C_{P_0} = P_0 \hat{h}^{R_0}, C_{R_0} = h^{R_0} \hat{h}^r_{R_0}$ を計算し，その AHO 署名のメッセージとして $(C_{x,0}, C_{T_0}, C_{P_0}, C_{R_0})$ を採用し，AHO 署名 $\sigma_0 = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7)$ をユーザに送付する．
  (4) ユーザ: ユーザの使用済みセッション ID の集合を $L_U = \emptyset$ とする．また $P_0 = h_1, R_0 = 0, r_{R_0} = 0$ とし，$C_{P_0} = P_0 \hat{h}^{R_0}, C_{R_0} = h^{R_0} \hat{h}^r_{R_0}$ を計算する．ユーザの証明書は以下となる．
  $cert_0 = (L_U, T_0, P_0, R_0, C_{x,0}, C_{T_0}, C_{P_0}, C_{R_0}, r_{x,0}, r_{T_0}, r_{R_0}, \sigma_0)$
  また，その秘密鍵を $sec = x$ とする．
- **認証**: 以下のプロトコルにより，ユーザは SP に自身の各カテゴリのスコアの合計値を示す．
  (1) SP: SP はユーザに L_S と全ての $j \in L_S$ に対する $s_{i,j}$ のリストを送付する．
  (2) ユーザ:証明書のコミットメント $(C_{x,t-1}, C_{T_{t-1}}, C_{P_{t-1}}, C_{R_{t-1}})$ を以下のようにラン

ダム化する．

- 乱数 $r'_{x,t-1}, r'_{T_{t-1}}, R'_{-1}, r'_{R_{t-1}}$ を選ぶ．
- $r''_{x,t-1} = r_{x,t-1} + r'_{x,t-1}, r''_{T_{t-1}} = r_{T_{t-1}} + r'_{T_{t-1}}, R''_{t-1} = R_{t-1} + R'_{t-1}, r''_{R_{t-1}} = r_{R_{t-1}} + r'_{R_{t-1}}$ とする．
- 以下に計算を行う．
  $C'_{x,t-1} = C_{x,t-1}\hat{h}^{r'_{x,t-1}} = h^x\hat{h}^{r''_{x,t-1}}$,
  $C'_{T_{t-1}} = C_{T_{t-1}}\hat{h}^{r'_{T_{t-1}}} = h^{T_{t-1}}\hat{h}^{r''_{T_{t-1}}}$,
  $C'_{P_{t-1}} = C_{P_{t-1}}\hat{h}^{r'_{P_{t-1}}} = h^{P_{t-1}}\hat{h}^{r''_{P_{t-1}}}$,
  $C'_{R_{t-1}} = C_{R_{t-1}}\hat{h}^{r'_{R_{t-1}}} = h^{R_{t-1}}\hat{h}^{r''_{R_{t-1}}}$
- $com_1 = (C'_{x,t-1}, C'_{T_{t-1}}, C'_{P_{t-1}}, C'_{R_{t-1}})$ とする．

(3) ユーザ: ユーザが所持している $cert_{t-1}$ の AHO 署名 σ_{t-1} をランダム化して $\sigma'_{t-1} = (\theta'_1, ..., \theta'_7)$ を得て，コミットメント $\{C_{\theta'_l}\}_{l\in(1,2,5)} = \theta'_l\hat{h}^{r_{\theta'_l}}$ を計算する．$com_{AHO_1} = (\{\theta'_l\}_{l\in(3,4,6,7)}, \{C_{\theta'_l}\}_{l\in(1,2,5)})$ とする．

(4) ユーザ: カテゴリのブロック $a(1 \le a \le \psi)$ に対して，以下を行う．まず，$S_a = \sum_{i=1}^{\phi}\sum_{k\in L_U} s_{i+(a-1)\cdot\phi,j}\cdot\ell_{i-1}$ を計算し，S_a に対する範囲の AHO 署名 σ_{a,S_a} をランダム化して $\sigma'_{a,S_a} = (\Theta'_{a,S_a,1}, ..., \Theta'_{a,S_a,7})$ を得る．そして，コミットメント $\{C_{\Theta'_{a,S_a,l}}\}_{l\in(1,2,5)} = \Theta'_{a,S_a,l}\hat{h}^{r_{\Theta'_{a,S_a,l}}}$ を計算する．$com_{AHO_{2,a}} = (\{\Theta'_{a,S_a,l}\}_{l\in(3,4,6,7)}, \{C_{\Theta'_{a,S_a,l}}\}_{l\in(1,2,5)})$ とする．

(5) SP: SP は全てのブロック $a(1 \le a \le \psi)$ に対して **AccGen*** より $acc_{L_S,a} = \prod_{i=1}^{\phi}\prod_{j\in L_S} g_{n+1-j}^{s_{i+(a-1)\cdot\phi,j}\cdot\ell_{i-1}}$ を計算し，ユーザに送付する．

(6) ユーザ: ユーザは **AccWitGen*** より $W_a = \prod_{i=1i}^{\phi}\prod_{k\in L_U}\prod_{j\in L_S}^{j\neq k} h_{n+1-j+k}^{s_{i+(a-1)\cdot\phi,j}\cdot\ell_{i-1}}$ を計算する．実際には，以前使用した補助情報 W'_a, リスト L_S との差分 δ_{L_S} に対して，$W_a = W'_a \cdot \prod_{i=1}^{\phi}\prod_{k\in L_U}\prod_{j\in\delta_{L_S}}^{j\neq k} h_{n+1-j+k}^{s_{i+(a-1)\cdot\phi,j}\cdot\ell_{i-1}}$ を計算すればよい．$r_{W_a} \in Z_p$ をランダムに選び，コミットメント $C_{W_a} = W_a\hat{h}^{r_{W_a}}$ を計算する．

(7) ユーザ: $R_t = R''_{t-1} = R_{t-1} + R'_{t-1}$ とする．$r_{R_t} \in Z_p$ をランダムに選び，$C_{R_t} = h^{R_t}\hat{h}^{r_{R_t}}$ を計算する．同様に T_t, r_{T_t} をランダムに選び，$C_{T_t} = h^{T_t}\hat{h}^{r_{T_t}}$ を計算する．$com_2 = (C_{W_1}, ..., C_{W_a}, C_{R_t}, C_{T_t})$ とする．

(8) ユーザ: SP へ $(com_1, com_2, com_{AHO_1}, com_{AHO_{2,1}}, ..., com_{AHO2,\psi}), T_{t-1}$ を送付する．

(9) ユーザ: SP へ以下の知識の署名 SPK を計算して送付する．

$SPK\{(x, T_{t-1}, R_{t-1}, r''_{x,t-1}, r''_{T_{t-1}}, r''_{R_{t-1}}, r_{R_t}, r'_{x,t-1}, r'_{T_{t-1}}, r'_{R_{t-1}}, R'_{t-1}, r'_{R_{t-1}}, r_{\theta'_1}, r_{\theta'_2}, r_{\theta'_5}, r_{\Theta'_{1,S_1,1}}, ..., r_{\Theta'_{\psi,S_\psi,1}}, r_{\Theta'_{1,S_1,2}}, ..., r_{\Theta'_{\psi,S_\psi,2}}, r_{\Theta'_{5,S_1,5}}, ..., r_{\Theta'_{\psi,S_\psi,5}}, r_{W_1}, ..., r_{W_\psi}) :$

$$C'_{x,t-1} = h^x\hat{h}^{r''_{x,t-1}} \bigwedge C'_{T_{t-1}} = h^{T_{t-1}}\hat{h}^{r''_{T_{t-1}}}$$
$$\bigwedge C'_{R_{t-1}} = h^{R_{t-1}}\hat{h}^{r''_{R_{t-1}}} \bigwedge C'_{R_t} = h^{R_{t-1}+R'_{t-1}}\hat{h}^{r_{R_t}}$$

$$\begin{aligned}&\bigwedge A^{-1}e(G_z, C_{\theta'_1})e(G_r, C_{\theta'_2})e(\theta'_3, \theta'_4)e(G_1, C'_{x,t-1})\\ &\quad e(G_2, C'_{T_{t-1}})e(G_3, C'_{P_{t-1}})e(G_4, C'_{R_{t-1}})\\ &= e(G_z, \hat{h})^{r_{\theta'_1}}e(G_r, \hat{h})^{r_{\theta'_2}}e(G_1, \hat{h})^{r'_{x,t-1}}\\ &\quad e(G_2, \hat{h})^{r'_{T_{t-1}}}e(G_3, \hat{h})^{R'_{t-1}}e(G_4, \hat{h})^{r'_{R_{t-1}}}\end{aligned} \tag{4.3}$$

$$\begin{aligned}&\bigwedge B^{-1}e(H_z, C_{\theta'_1})e(H_r, C_{\theta'_5})e(\theta'_6, \theta'_7)e(H_1, C'_{x,t-1})\\ &\quad e(H_2, C'_{T_{t-1}})e(H_3, C'_{P_{t-1}})e(H_4, C'_{R_{t-1}})\\ &= e(H_z, \hat{h})^{r_{\theta'_1}}e(H_r, \hat{h})^{r_{\theta'_5}}e(H_1, \hat{h})^{r'_{x,t-1}}\\ &\quad e(H_2, \hat{h})^{r'_{T_{t-1}}}e(H_3, \hat{h})^{R'_{t-1}}e(H_4, \hat{h})^{r'_{R_{t-1}}}\end{aligned} \tag{4.4}$$

$$\begin{aligned}&\bigwedge_{a=1}^{\psi} e(acc_{L_S,a}, C'_{P_{t-1}})e(g, C_{W_a})^{-1}\\ &= e(acc_{L_S,a}, \hat{h})^{R_{t-1}+R'_{t-1}}e(g, \hat{h})^{-r_{W_a}}e(g_1, g_n)^{S_a}\end{aligned} \tag{4.5}$$

$$\begin{aligned}&\bigwedge_{a=1}^{\psi} \hat{A}^{-1}e(\hat{G}_z, C_{\Theta'_{a,S_a,1}})e(\hat{G}_r, C_{\Theta'_{a,S_a,2}})\\ &\quad e(\Theta'_{a,S_a,3}, \Theta'_{a,S_a,4})\\ &= e(\hat{G}_z, \hat{h})^{r_{\Theta'_{a,S_a,1}}}e(\dot{G}_r, \hat{h})^{r_{\Theta'_{a,S_a,2}}}e(\hat{G}_1, \hat{h})^{-S_a}\end{aligned} \tag{4.6}$$

$$\begin{aligned}&\bigwedge_{a=1}^{\psi} \hat{B}^{-1}e(\hat{H}_z, C_{\Theta'_{a,S_a,1}})e(\hat{H}_r, C_{\Theta'_{a,S_a,5}})\\ &\quad e(\Theta'_{a,S_a,6}, \Theta'_{a,S_a,7})\\ &= e(\hat{H}_z, \hat{h})^{r_{\Theta'_{a,S_a,1}}}e(\dot{H}_r, \hat{h})^{r_{\Theta'_{a,S_a,5}}}e(\hat{H}_1, \hat{h})^{-S_a}\}\end{aligned} \tag{4.7}$$

(10) SP: 送られた T_{t-1} が過去に使用されていたものだった場合は認証失敗とする．使用されていなければ過去に SP へ送られているタグの集合 S に加える．

(11) SP: このユーザが使用するセッション ID j を選択する．このユーザのアキュームレータにセッション ID を加えるために $C_{P_t} = C_{P_{t-1}}h_j$ を計算する．これにより $P_t = P_{t-1}h_j$ と更新される．

(12) SP: $C_{x,t} = C'_{x,t-1}$ としてメッセージ $(C_{x,t}, C_{T_t}, C_{P_t}, C_{R_t})$ に対して AHO 署名 σ_t を作成する．その後，(σ_t, j) をユーザに送付する．

(13) ユーザ: L_U に j を加える．$P_t = P_{t-1}h_j$ を計算する．これにより $\prod_{j\in L_U} h_j$ となる．

表 1　実装環境

OS	Ubuntu 16.04 LTS 64bit
CPU	Intel®Core™i5-8400 CPU@2.80Ghz
メモリ	3.9GB
ペアリングライブラリ	PBC-0.5.1.4

(14)ユーザ: ユーザはセッション j と以下の新しい証明書を得る．

$cert_t = (L_U, T_t, P_t, R_t, C_{x,t}, C_{T_t}, C_{P_t}, C_{R_t}, r_{x,t}, r_{T_t}, r_{R_t}, \sigma_t)$

4.6　安全性

(4.3) 式を整理すると，$\Theta'_1 = C_{\Theta'_1}\hat{h}^{-r_{\Theta'_1}}, \Theta'_2 = C_{\Theta'_2}\hat{h}^{-r_{\Theta'_2}}, C_{x,t-1} = C'_{x,t-1}\hat{h}^{-r'_{x,t-1}}, C_{T-1} = C'_{T-1}\hat{h}^{-r'_{T-1}}, C_{P-1} = C'_{P-1}\hat{h}^{-R'_{T-1}}, C_{R_{t-1}} = C'_{R_{t-1}}\hat{h}^{-r'_{R_{t-1}}}$ に対して，

$A = e(G_z, \theta'_1)e(G_r, \theta'_2)e(\theta'_3, \theta'_4)e(G_1, C_{x,t-1})e(G_2, C_{T_{t-1}})$
$e(G_3, C_{P_{t-1}})e(G_4, C_{R_{t-1}})$

となり，コミットメント $(C_{x,t-1}, C_{T_{t-1}}, C_{P_{t-1}}, C_{R_{t-1}})$ に対する AHO 署名の検証式が得られる．(4.4) 式も同様である．さらに (4.5) 式から，以下が得られる．

$$\frac{e(acc_{L_S,a}, C_{P_{t-1}}\hat{h}^{-(R_{t-1}+R'_{t-1})})}{e(g, C_{W_a}\hat{h}^{-r_{W_a}})} = e(g_1, g_n)^{S_a}$$

$W_a = C_{W_a}\hat{h}^{-r_{W_a}}, P_{t-1} = C_{P_{t-1}}\hat{h}^{-(R_{t-1}+R'_{t-1})}$ とすると，

$$\frac{e(acc_{L_S,a}, P_{t-1})}{e(g, W_a)} = e(g_1, g_n)^{S_a}$$

となり，アキュームレータの検証式が得られる．

次に (4.6) 式を整理すると，(4.3) 式と同様に

$\hat{A} = e(\hat{G}_z, C_{\Theta'_{a,S_a,1}})e(\hat{G}_r, C_{\Theta'_{a,S_a,2}})e(\Theta'_{a,S_a,3}, \Theta'_{a,S_a,4})$
$e(\hat{G}_1, \hat{h}^{S_a})$

となり，$\hat{h}^{S_a}$，すなわち S_a に対する AHO 署名の検証式が得られる．(4.7) 式も同様である．

匿名性: 各情報はコミットメント，SPK によって秘匿された形で送信されるので，ユーザを特定することはできない．また，同様に同じユーザの認証かどうかも分からない．さらに，評価値の和も範囲しか分からないため，それによる追跡もできない．

偽造不能性: 登録と認証では，AHO 署名である証明書の保持が証明される．AHO 署名の偽造不能性から未登録ユーザは認証に失敗する．またアキュームレータの安全性より，そのユーザの総評価値 $S_a = \sum_{i=1}^{\phi}\sum_{k\in U}\sum_{j\in V}^{j\neq k} s_{i+(a-1)\cdot\phi,j} \cdot \ell_{i-1}$ が保証される．そして範囲証明の AHO 署名の保持も証明されることから，範囲も保証される．

5.　実装評価

5.1　実装環境と計測対象

本研究では表 1 に示す環境で実装を行なった．また評価対象は，カテゴリを変化させたときの，認証における処理時間と SP とユーザが保持す範囲の AHO 署名の全データサイズとする．さらにそれぞれの評価対象において，単純にポリシーを導入した方式，全てのカテゴリでビット結合を行なった方式，バランス化した提案方式について比較を行う．なお，署名のデータサイズは，取りうる評価値の和の有効な値の数に依存しているが，評価値の和の有効な値は，$-50 \sim +50$，$-25 \sim 25$ の場合を示す．

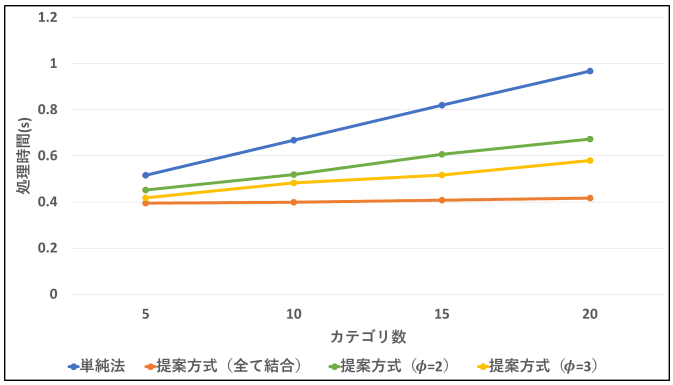

図 1　認証の処理時間
Fig. 1　authentication time

表 2　$-50 \sim +50$ までの署名のデータサイズ（単位:MByte）

カテゴリ数	5	10	15	20
単純法	1.1	2.2	3.3	4.4
提案方式（全て結合）	2.3×10^{7}	2.42×10^{17}	2.54×10^{27}	2.67×10^{37}
提案方式 ($\phi=2$)	6.8×10	3.34×10^{2}	6.3×10^{2}	1.23×10^{3}
提案方式 ($\phi=3$)	2.31×10^{3}	1.36×10^{4}	3.4×10^{4}	4.76×10^{4}

表 3　$-25 \sim +25$ までの署名のデータサイズ（単位:MByte）

カテゴリ数	5	10	15	20
単純法	0.56	1.12	1.68	2.24
提案方式（全て結合）	7.59×10^{5}	2.62×10^{14}	9.04×10^{22}	3.12×10^{31}
提案方式 ($\phi=2$)	1.75×10	8.58×10	1.61×10^{2}	3.14×10^{2}
提案方式 ($\phi=3$)	3.03×10^{2}	1.75×10^{3}	4.38×10^{3}	6.17×10^{3}

5.2　計測結果

取りうる評価値の和が $-50 \sim +50$ の時の認証の処理時間の計測結果は**図 1** である．$-25 \sim +25$ の場合も同様である．署名のデータサイズは**表 2**，**表 3** の通りである．単純法では処理時間がカテゴリ数に比例して増加するが，署名のデータサイズは 8.8Mbyte 程度となり管理が容易なサイズとなっている．全てのビットを結合する提案方式では処理時間はカテゴリ数に依らずほぼ一定であるが，署名のデータサイズはカテゴリ数が 5 の時で 2.3TByte となり膨

大なものとなっている．改良方式では処理時間の増加が抑えられており，また署名のデータサイズもカテゴリ数が 20 の時で $\phi = 2$ であれば 1.23GB，$\phi = 3$ であれば 47.6GB となり，提案方式と比較して小さくなっていることが分かる．

5.3 評価

改良方式において，パラメータ ϕ は 2 もしくは 3 としている．評価値の和の取りうる値が $-50 \sim +50$ の場合，カテゴリ数が 5 において，署名のデータサイズはそれぞれ 68MB，2.31GB となった．$\phi = 3$ の場合はすでに 1GB を超えているため，カテゴリ数が少ない場合でも署名データの管理が難しい．またカテゴリ数が 20 の時では，$\phi = 2$ の場合でも 1GB を超えてしまう．しかし評価値の和の取りうる値が $-25 \sim +25$ の場合は $\phi = 2$ であればカテゴリ数が 20 の時でも 314MByte となっており，1GB を越えないため管理が可能であるといえる．

6. まとめ

本研究では，従来のブラックリスト型匿名認証システムにポリシーを導入した方式を提案した．全てのカテゴリについてビット結合を行うと，範囲証明の署名のデータサイズが膨大なものなるため，全カテゴリを ϕ 個ごとにブロック化しそのブロックごとにビット結合を行うよう改良する方式を提案した．この改良により認証の処理時間と署名のデータサイズのバランスをパラメータ ϕ により調節できるようになった．

また実装の結果，$\phi = 2$ とした場合に認証時間は単純法の半分の時間に削減され，署名のデータサイズはカテゴリ数が 15 までは管理が容易なものであることが分かった．

今後の課題として，ポリシーの論理式に OR を導入することが考えられる．

謝辞 本研究の一部は，JSPS 科研費 (JP19K11964) の助成を受けている．

参考文献

[1] P.P.Tsang, M.H.Au, A.Kapadia, S.W.Smith, ”Blacklistable Anonymous Credentials: Blocking Misbehaving Users without TTPs”, ACM-CCS2007, pp.72-81, 2007.

[2] Y.Aikou, S.Sadiah, T.Nakanishi, ‘‘An Efficient Blacklistable Anonymous Credentials without TTPs Using Pairing-Based Accumulator’’, IEEE-AINA2017, pp.780-786, 2017.

[3] M.H.Au, A.Kapadia, W.Susilo, ”BLACR: TTP-Free Blacklistable Anonymous Credentials with Reputation”, NDSS Symposium 2012, pp.1-17, 2012.

[4] 金谷健士, 中西透, ‘‘アキュームレータを用いた評価値ベースのブラックリスト型匿名認証’’, 信学技報, vol.117, no.369, ISEC2017-81, pp.59-65, 2017.

[5] 金谷健士, 中西透, ‘‘アキュームレータを用いた評価値ベースのブラックリスト型匿名認証の拡張’’, CSEC, 2018-CSEC-83(15), pp.1-7, 2018.