平成21年度

卒業論文

研究記録管理・公開・検証システム arχves への確認者署名機能の付加

指導教員　小林　聡　准教授

島根大学　総合理工学部
数理・情報システム学科　情報工学コース
S043028　加藤康

目次

1 はじめに

情報化社会である現代において、研究記録を電子媒体に保存することは当然の考えであるが、電子媒体に保存されている研究記録は、紙媒体に保存されている研究記録と比較し、記録の改竄が行われる危険性が高いことは容易に想像できるであろう。電子媒体に保存された研究記録の改竄の発生を防止することは技術的に困難であるが、改竄が困難な、あるいは改竄の検出が容易な手法によって研究記録が保存されているならば、研究記録の検証に関しては大いに助けになると考えられる。この際、研究記録の正真性と非改竄性の保証をいかにして行うかが課題となる。

このような課題に対して、原田らは電子カルテを念頭に、公開鍵暗号を用いて、記述者が自分自身の秘密鍵で電子署名をすると共に、文書管理システムに持たせた秘密鍵を使った電子署名により検印を行うモデルの提案を行っている[1]。また、陳らは XML 文書の時刻認証を伴った管理を試みている[2]。同様なサービスとして、SNS/Blog 機能に電子文書へのタイムスタンプ付加機能を加えたサービスも存在する[*1]。しかし、これは日本電子公証機構のサービスを利用しているため、比較的高額なサービスとなっている。このような、タイムスタンプを活用したサービスは DVCS（Data Validation &Certification Server Protocol）や TAP（Trusted Archival Protocol）などに分類される。DVCS は、電子文書のアーカイブは行わないことを原則としており、対して TAP はアーカイブも行うことを原則としている。また宇根らは DVCS や TAP で用いられる各種タイムスタンプによる可用性・安全性に関して報告している[3]。

そして記録、コミュニケーションと並ぶ、コンピュータが果たすことができる重要な役割として、ストレージ機能がある。研究資料や実験試料、あるいはプログラムなどを手軽に公開・参照できる機能が実現できれば、研究者間での研究資料や実験試料などの流通も盛んとなり、研究自体が活性化するであろう。

ただし、記録機能においても、ストレージ機能においても、研究の遂行途中においては一般には秘匿したい、あるいは秘匿する必要のある情報も存在する。そのため、情報を公開する範囲を柔軟に制御できなければならない。

現在、World Wide Webを用いたSNS(Social Network Service)などのサービスにおいて、記事の公開範囲はある程度の制御が可能ではあるが、概ねその設定の自由度は低い。

Masui らや江渡らは、QuickML[4]やqwikWeb[5]により、情報にアクセスできる者を柔軟に変更可能とするシステムの構築と運用を試みた。また、SNSを基盤としたシステムとして、永田らはEnzinを[7]、高井らはACSを開発し[6]、運用実験を行った。高田らは、公開Web-DB とWeb-DB管理システムを分離可能であり、かつきめ細かなアクセス制御が可能なWeb-DB管理システムを構築している[8]。また、一般のSNS/blogにも、記事ごとに公開範囲の設定を可能とするサービスが登場しており[*2]、公開範囲の柔軟な制御の需要が伺える。

加藤らは、上記のような、研究記録などの管理を主目的とし、公開範囲を柔軟に設定可能かつ、記録の正真性および非改竄性を可能な限り保証することを目的とした研究記録管理・公開・検証システム ar χ ves[*3][9、10、11]の構築を提案した。

*1 Synest LaboNote（http://www.labonote.jp）

*2 Media Wagon(株式会社エイミー http://mw.aimy.jp)

*3 arχves: Archive system for Research logs by Kato[2]=Shima-nuki/Kobayasi Satoshi， and VErification System

ar χ ves は、公開鍵暗号を用いたディジタル署名を記事に対して二重に行うことで、記事の正真性と非改竄性を保証を行っている。二重署名の一つは投稿者本人による署名(ユーザ署名)であり、もう一つは、ユーザ署名の行われた記事に対して行う、第三者の署名(サーバ署名)である。記事に対して、上記の二重署名処理を行うことによって、記事の正真性と非改竄性の保証が行われている。

現在、米国では、特許取得の際、先発明主義[*4]という方針に沿っている。発明日の立証にはラボノートによる研究記録の記載が重要になるが、ラボノートが特許取得の際の証拠として使用できる条件として、記録の改変が不可能な事及び、第三者による確認と署名などが挙げられる。

ar χ ves におけるサーバ署名は、サーバによって機械的に行われた署名である。その為、現状では記事本文の内容を理解できる者によって署名が行われていない。そこで、ar χ ves によって保存されている記録の知的財産的価値を高める為には、確認者(or 証人)による記事の確認、署名が必要であると考えられる。

本研究では、研究記録を管理、検証、公開するシステムである ar χ ves に、確認者(or 証人)による署名を付加し、研究記録の知的財産としての価値を高めることを目標とした改良を行った。

→ ARKSVES. KSをxに、さらにxを類字形のχに置き換え。

*4 先発明主義: 最初に発明した発明者に特許権を与える制度

2 システムの特徴

2.1 概要

本システムは、研究記録などの管理を主目的とし、公開範囲を柔軟に制御可能かつ、記録の正真性及び非改竄性を可能な限り保証することを目的としたシステムである。

データの正真性と非改竄性の保証を可能とするため、記事データ投稿時に、記事データに対して二重署名の処理を行い、平易な操作で検証を行えるようしている。また、電子情報の簡便な記録・保存と扱いやすさのために、基本的な操作は近年広く普及しているSNS/Blogをベースとしている。そして、柔軟な公開範囲の実装の為、一般のSNS/Blogのように Blogコンテンツ全体の公開範囲を指定する方式ではなく、投稿される記事ごとに公開範囲を指定し、公開範囲の設定を自由度の高い指定を可能としている。

特徴を要約すると以下4点となる。

- 正真性と非改竄性の保証を可能とする電子データに対する署名及び、検証機能
- SNS/Blogをベースとした実装
- 記事毎の柔軟な公開範囲の指定が可能

以降、各節で機能の詳細について説明を行う。

2.2 利用ツール

本システムで使用しているツール等を表-1に示す。

Ruby on Railsは本システムの基幹となるSNS/Blog機能の構築を中心に活用し、データベースにはMySQLを使用した。電子署名の付与及び検証は、全てGnuPGを活用している。

表-1 利用ツール一覧

開発言語：	Ruby ver. 1.8.6
フレームワーク：	Ruby on Rails ver. 1.2.6
公開鍵暗号ソフト：	GnuPG ver. 1.4.9
データベース：	MySQL ver. 5.0.27
ウェブサーバ：	Apache ver. 2.0.6
	Mongrel Web Server ver. 1.1.5
SSLライブラリ：	OpenSSL ver. 0.9.8

2.3 二重署名による記事の正真性と非改竄性の保証

企業における、記録の正真性及び、非改竄性を保証する方法として、部下の作成した記録に対し、上司が検印を捺すことが広く行われている。このような、記録の記述者とは異なる者による検印、あるいはそれに類似の行為/操作を行うモデルが原田らによって提案されており、そのようなモデルをここでは「検印モデル」と呼ぶ。

本システムは上記の検印モデルを元に、記事の正真性と非改竄性の保証を行う。記事やデータに検印を行うサーバと文書を保管するサーバの異なる2つのサーバを用意し、公開鍵暗号を用い、記述者が十分自身の秘密鍵で電子署名をするとともに、検印サーバの秘密鍵で電子署名を

行うことで、検印モデルを実現している。具体的な記事投稿後のシステムの動作を以下①～⑤に示す。

① 投稿された記事、データに対して、文書サーバがタイムスタンプを付すとともに、ユーザの秘密鍵で署名を行う。
② ①で作成されたデータを検印サーバへ転送する。
③ 文書サーバから送られたデータに対し、検印サーバがタイムスタンプを付し、更にデータ全体に対し署名を行う。その後、署名部のみを検印サーバに保存する。
④ 検印サーバは文書サーバに2重に署名済みの記事、データを送り返す。
⑤ 文書サーバに2重署名済みデータを保存する。

以上の一連の流れを図式化したものを図1に示す。また、2重署名の行われた実際の文書の例を図2に示す。

文書サーバと検印サーバ間での通信には、セキュアな通信の為 SSL を用いた通信を行っている。

また、文書サーバと検印サーバのそれぞれで行われる署名の直前に、データに対して時刻情報を書き込み、タイムスタンプのシンプル・プロトコルと同様な作業を行っている。

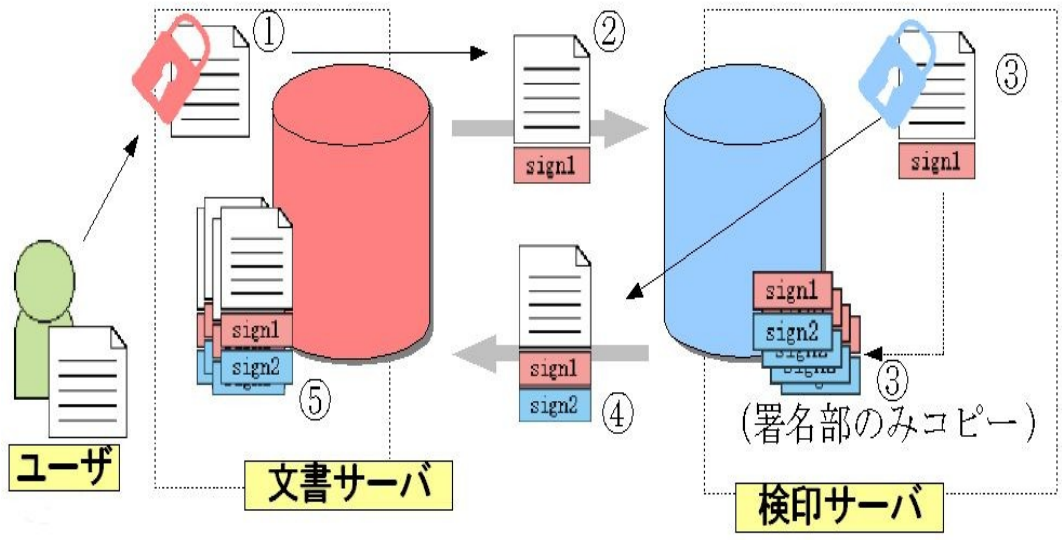

図1　2重署名処理の流れ

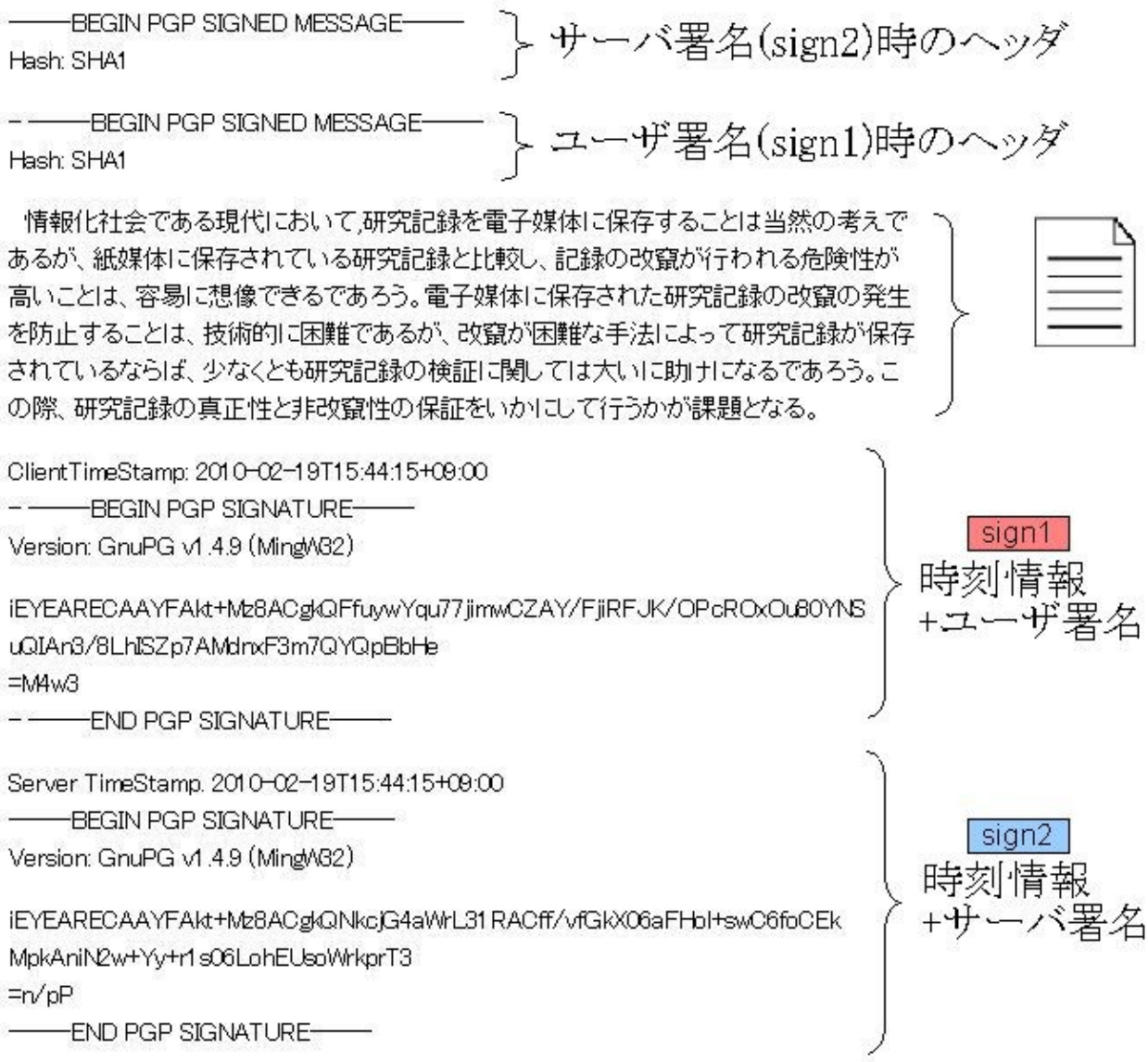

図２ ２重署名された記事データの構造

2.4 記事及び署名ファイルに対する検証機能

前節の記事データに対して行った、二重のデジタル署名によって、投稿されたデータの正真性および非改竄性の保証が行われているが、それが確かであるかどうかは署名の検証作業を行う必要がある。

また、電子商取引推進協議会、認証・公証WG [15]のガイドラインでは、デジタル署名の有効性を長期的に維持する為の用件として、次の4つを挙げている。

1. 署名検証時に、署名再検証に必要な情報を明確にしておくこと
2. 署名検証時の時刻を明確にしておくこと
3. 署名再検証時に必要な情報を改竄検出可能な状態にすること。
4. 署名再検証時に必要な情報を保存すること

（1）は、検証を行う際に確かにその署名が有効であることを示す情報（あるいは失効情報）を明示しておくこと。そして、署名の検証が確かに保証されていることを確認した日時を明確にし、（2）その時点まで確かにデータの保証されていることを記録し、（3）再検証を正しく行えるように改竄をの検出を可能にして、（4)署名の再検証に必要な情報を保存し、第三者でも再検証を行えるようにしておくということである。これらの用件を満たすことがデジタル署名の有効性の長期的な維持には必要であるとされている。

このように、本当にその署名が確かであることを確認したり、あるいは長期的な有効性の維持には「署名の検証」が必要となってくる。本システムでは、その検証を簡単に行う事ができる機能を実装している。

各記事およびデータには、「簡易検証」機能および「通常検証」機能が用意され、閲覧者はそれらのリンクをクリックすることで簡便な検証を行うことができる。なお、「簡易検証」は文書サーバのみで、ユーザの公開鍵と検印サーバの公開鍵を用いて行う検証である。通常検証は、検印サーバに記録されている署名データとの照合も含めて検証を行う。検証を行なった画面例を図3に示す。また、記事の内容のみを書き換えられた場合の検証画面例を図4に示す。

検証結果

投稿者:管理者

検証記事タイトル:arχvesについて

投稿日時:2010-02-19 15:44:15

ユーザ主鍵フィンガープリント:
F4BB BDCE 6667 7ACB 60AE AC0B 15FB B2C1 8AAE EFB8

検証結果:確認者サーバー署名

検証結果:確認者署名

検証結果:サーバー署名

gpg: 02/19/10 15:44:15にDSA鍵ID 8696ACBDで施された署名
gpg: "server (sarver's key)"からの正しい署名

検証結果:ユーザ署名

gpg: 02/19/10 15:44:15にDSA鍵ID 8AAEEFB8で施された署名
gpg: "MikuKatou "からの正しい署名
gpg: 警告: この鍵は信用できる署名で証明されていません!
gpg: この署名が所有者のものかどうかの検証手段がありません。
主鍵の指紋: F4BB BDCE 6667 7ACB 60AE AC0B 15FB B2C1 8AAE EFB8

検印サーバ側の署名部との照合結果

検印サーバ側の署名部と一致しています。
・全ての署名が一致しています。

図3 検証結果(非改竄時)

検証結果

投稿者:管理者

検証記事タイトル:arχvesについて

投稿日時:2010-02-19 15:44:15

ユーザ主鍵フィンガープリント:
F4BB BDCE 6667 7ACB 60AE AC0B 15FB B2C1 8AAE EFB8

検証結果:確認者サーバー署名

検証結果:確認者署名

検証結果:サーバー署名

gpg: 02/19/10 15:44:15にDSA鍵ID 8696ACBDで施された署名
gpg: "server (sarver's key)"からの 不正な 署名

検証結果:ユーザ署名

gpg: 02/19/10 15:44:15にDSA鍵ID 8AAEEFB8で施された署名
gpg: "MikuKatou "からの 不正な 署名

検印サーバ側の署名部との照合結果

検印サーバ側の署名部と一致しています。
・全ての署名が一致しています。

図4 検証結果(記事改竄時)

2.5 SNS/Blogベース

2.5.1 概要

本システムは、電子情報の簡便な記録・保存と扱いやすさの為、基本的な操作は近年広く普及しているSNS/Blogをベースに実装している[19]。

本システムにアクセスすると、図5のようなトップページコンテンツが表示される。ここでユーザとして認証を行い、ログインを行うと、図6のような画面が表示される。

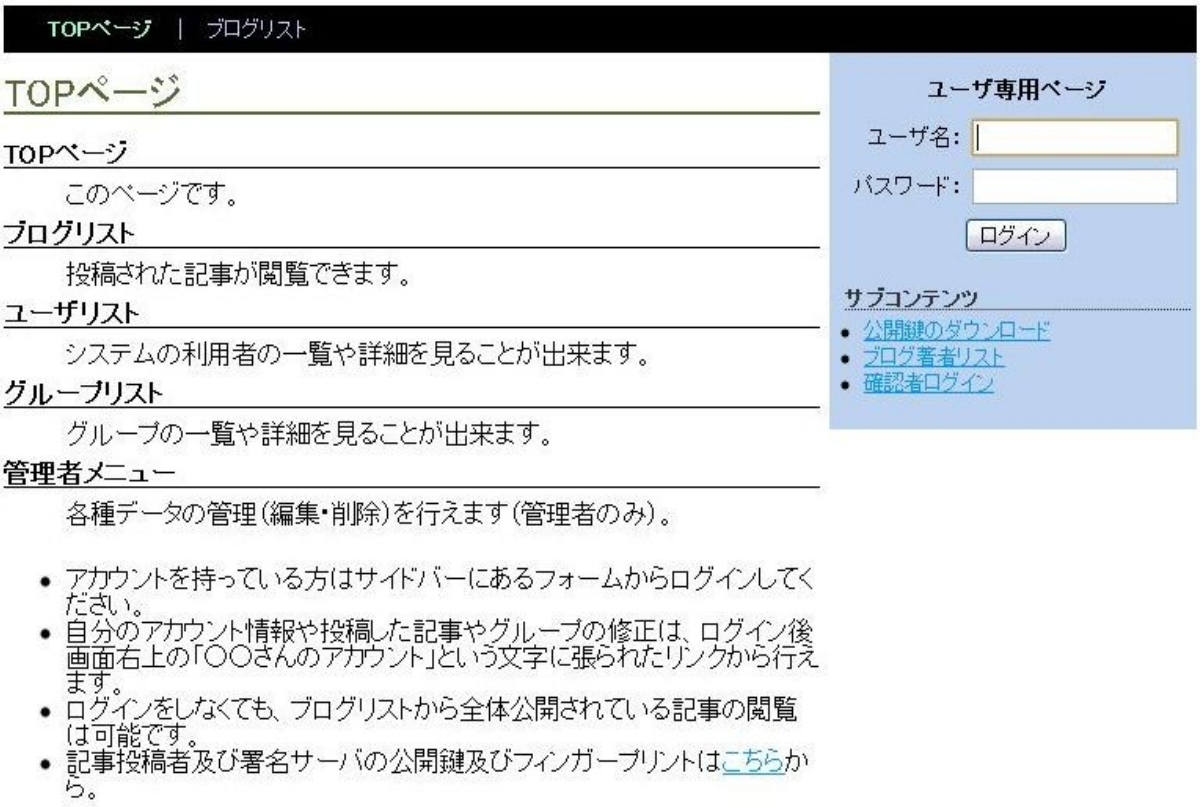

図5 未ログイン時トップページ

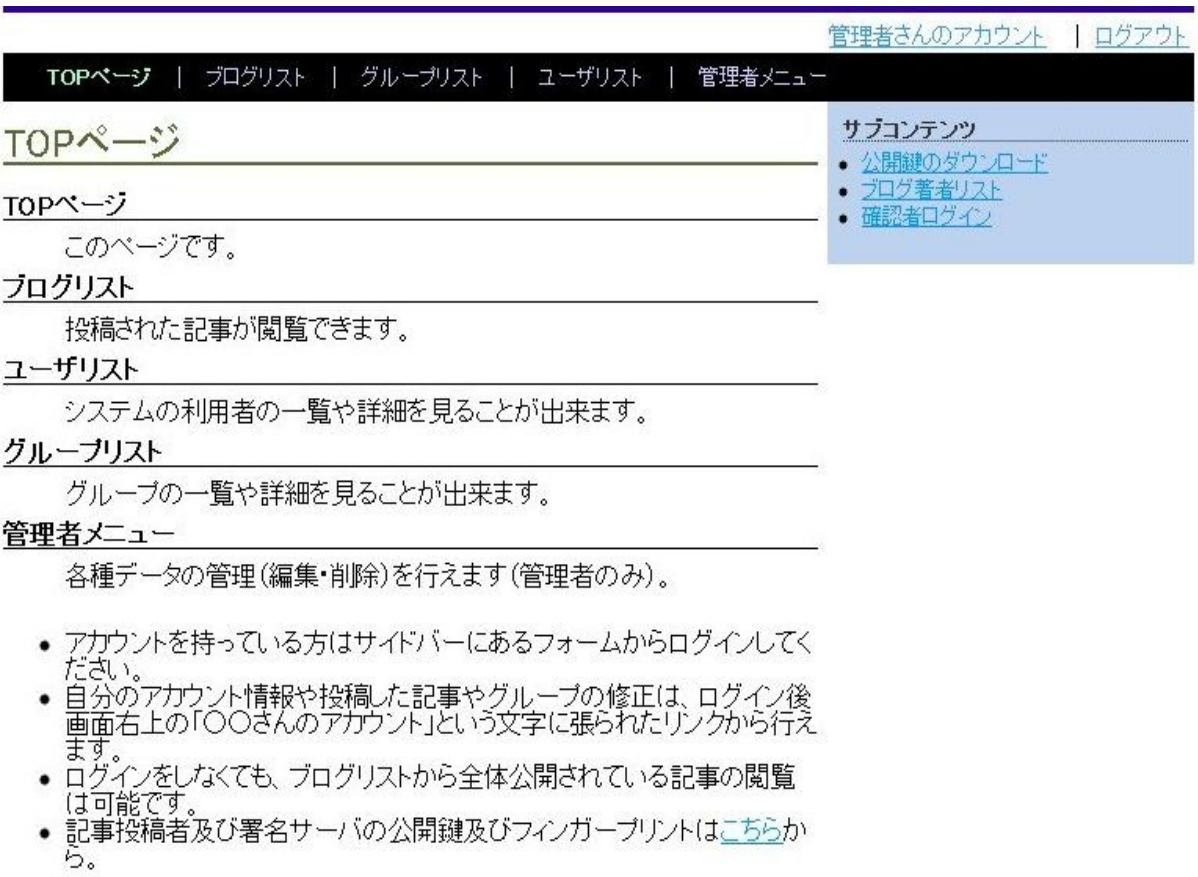

図6　ログイン後トップページ

ページは大きくメニューバー、メインページ、サイドバーで構成されている(図7)。

メニューバーには各コンテンツへのリンク、メインページには各コンテンツの内容、サイドバーにはコンテンツ内のサブコンテンツ/関連ページへのリンクが表示される。

また、ログインすることで、メニューバー上部にアカウント名が表示され、各アカウントメニューへのリンク及びログアウト用のリンクが表示される。メニューバー及びサイドバーは、ログインしたアカウントの権限ごとに異なったコンテンツが表示される。

以下でコミュニティを形成する利用者及び、グループ、利用者による記事の投稿をメインとした交流と、その管理を行う為のコンテンツ毎の機能について示す。

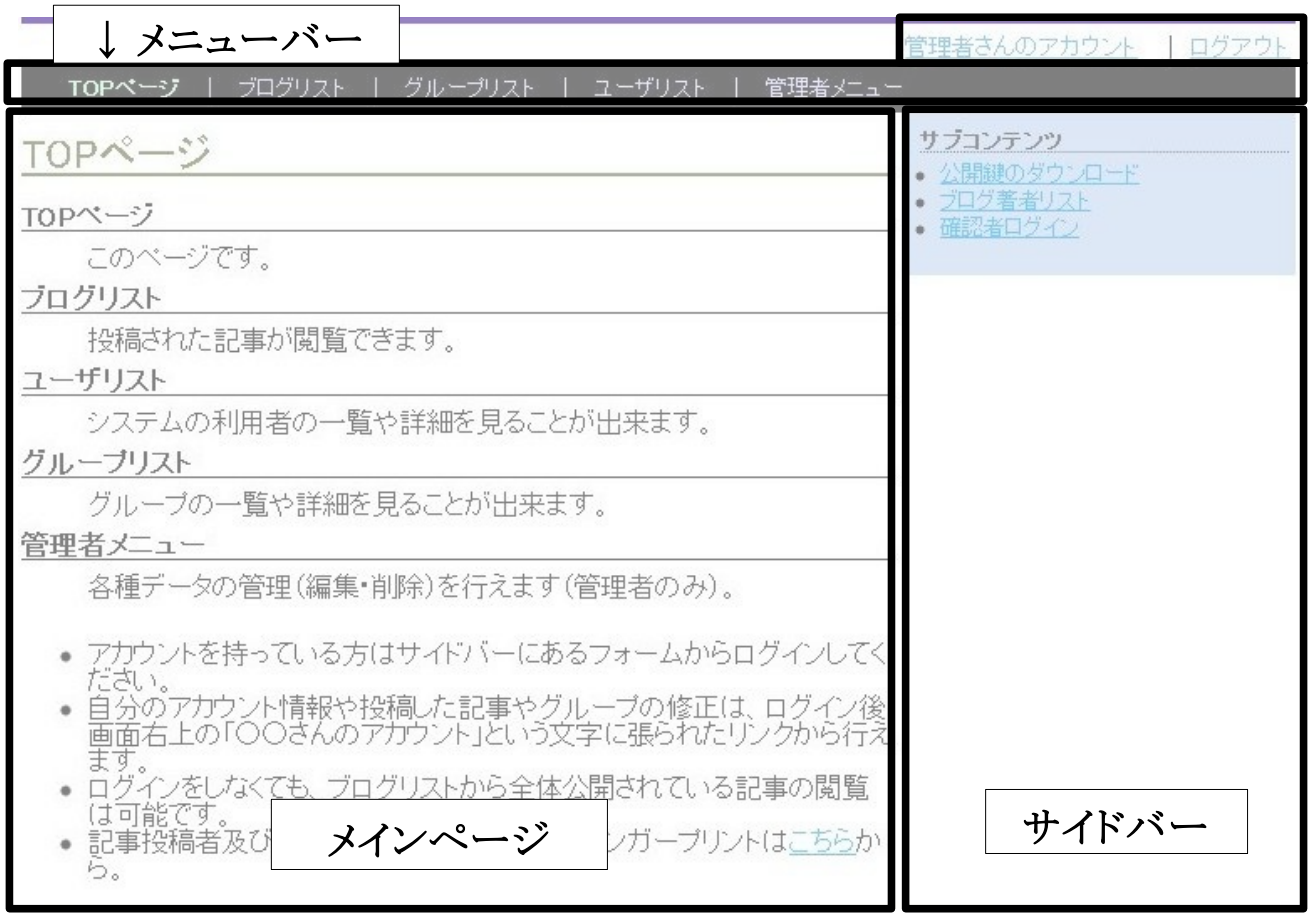

図7　画面構成

2.5.2 利用者の分類

本システムの全ての利用者は、本システム上に何らかのアカウントを所持している。利用者は、システムを利用できる権限によって、「ゲスト」、「ユーザ」、「管理者」に分類される。利用者ごとの大まかなアクセス制限を表2に示す。

「ゲスト」は、システム内のblog記事の投稿や、グループの作成は行うことが出来ないが、グループに所属し、記事の閲覧は可能な利用者である。

「ユーザ」は、システム内に自身のblogを作成し、記事を投稿することができる利用者で、グループの新規作成や、自身のアカウント情報を修正することが可能である.、また、自身が投稿した記事の各々について、公開範囲の変更や、カテゴリの変更を行うことができる。

「管理者」は、システムの管理者であり、管理者メニューにアクセス出来る、唯一の利用者である。

本システムはトップページ上のログインフォームから、IDとパスワードによる、ユーザ認証を受け、ログインを行うことで、利用者の権限に応じたコンテンツが表示される。また、システムにログインしていなくても、公開範囲に制限のない記事に関しては閲覧を行うことが可能である。

表中の、「アカウントメニュー」は、ログインした利用者自身の情報の編集、削除が行えるコンテンツである。また、「管理者メニュー」は、システム全体の各データの編集、削除が行えるコンテンツである。

表2　アクセス制限

コンテンツ ＼ 利用者	管理者	ユーザ	ゲスト	一般
トップページ	○	○	○	○
ブログリスト	○	○	○	○
グループリスト	○	○	○	
ユーザーリスト	○	○	○	
アカウントメニュー	○	○	△	
管理者メニュー	○			

2.5.3 グループについて

グループは、一人以上の利用者からなる、利用者の集合である。グループの種類は階層別に、「ルートグループ」、「トップレベルグループ」、「子グループ」に分けられる。(図8)

ルートグループは、アカウントを持つ全ての利用者が所属するグループである。そのため、利用者は、必ず一つ以上のグループに所属していることになる。

トップレベルグループは、ルートグループの直下に作成されるグループであり、利用者が自分で作成できる一番上位のグループである。

子グループは、トップレベルグループよりも下層に作成されるグループである。必ず親として、トップレベルグループか、子グループを一つ持つ。

以上より、グループに階層構造を持たせている。グループの種類と命名規則についての詳細を表3に示す。

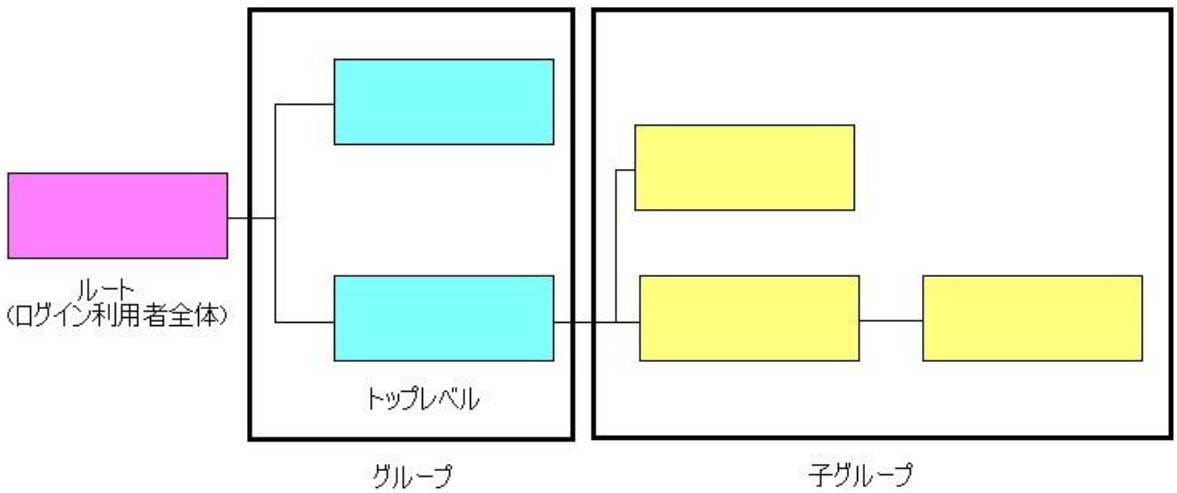

図8　グループの階層構造

表3 グループの種類と命名規則

グループの種類	階層	命名規則	新規作成
ルートグループ	0	::	不可
トップレベルグループ	1	::グループ名	可
子グループ	2～n	::トップグループ名:グループ名	可

2.5.4 利用者の管理(ユーザリスト)

システム利用者の情報の閲覧は、ログインしている利用者ならば、コンテンツの「ユーザリスト」から、誰でも参照することができる。また、トップページの名前に張られたリンクから、各自の詳細な情報を閲覧することも可能である。

ユーザの新規登録及び、修正、削除は、管理者メニューの「ユーザ管理」からのみ行うことが可能である。また、利用者の、「管理者」、「ユーザ」、「ゲスト」権限の指定も、管理者のみが設定可能である。

ただし、利用者自身の情報(アカウント情報)は、ユーザであれば一部の情報の変更が可能である。アカウント情報の変更は、アカウントメニューの「ユーザ情報の変更」から行うことができる。

2.5.5 グループの管理

グループリストの閲覧はログインしている利用者であれば、「グループリスト」から、誰でも閲覧することが可能である。

グループの新規作成は、ユーザ及び、管理者であれば、誰でも作成可能である。その際、新規作成できるグループは、トップレベルグループと、子グループの2つのグループがあり、どちらを作成するかを選択することができる。子グループは親となるグループのIDを指定することで作成することができる。また、子グループを親とする孫グループなどを作成することも可能である。

また、グループの管理は、グループを作成したユーザと、管理者が行うことができる。この二者を「グループ管理者」と呼ぶ。グループ管理者は、グループの編集、削除、参加者の変更を行うことが可能である。さらに管理者は上記に加えて、グループ作成者の変更が可能である。
各グループの詳細な情報は、トップページのグループ名のリンクより閲覧することができる。

2.5.6 記事の投稿・管理(ブログリスト)

記事の閲覧は、コンテンツの「ブログリスト」から行うことができる。コンテンツへのアクセスは誰でも可能だが、実際の閲覧については、記事ごとに指定された閲覧範囲によって制限がされている。つまり、公開範囲によっては、本システムへのログインを行っていなくても、記事の閲覧を行うことが可能になっている。閲覧者が公開範囲に含まれていない記事は、トップページにおいて、記事内容の代わりに、「閲覧権限がありません」と表示される。ただし、タイトルのみの閲覧、投稿日時の閲覧、記事の検証は行うことが可能である。

記事の新規作成は、ログインしているユーザと管理者であれば、「ブログリスト」コンテンツのサイドバーより行うことが可能である。実際の投稿画面を、図9に示す。(図9中の確認者選択項目については後述)

記事投稿の際の2重署名処理についての詳細は、2．3節で行った。2重署名を含めた記事の全文を表示したページを図10に示す。

また、各記事にはファイルを添付することが可能である。添付されたファイルについても、記事と同様に2重署名処理が行われる。添付ファイルのダウンロードは、2重署名の行われたファイルと、システムが自動的に復号処理を行ったファイルの両方について、記事の詳細ページより行うことが可能である。

なお、現在、投稿後の記事の修正・再編集は認めていない。記事の削除については、管理者のみが行うことが可能である。また、添付ファイルの削除は、投稿者も自身の記事のアカウントメニュー、「ブログ記事の閲覧範囲の変更/添付ファイルの削除」より、行うことが可能であるが、削除理由を明記した上で削除可能としている。また、添付データが存在していた記録が残るようにしている。これは、研究記録の非改竄性を考慮してのことである。このように編集、削除に制限を行っているが、記事などの管理にはDBMSを利用しているので、DBMSを直接操作することにより、記事の修正、削除は可能である。

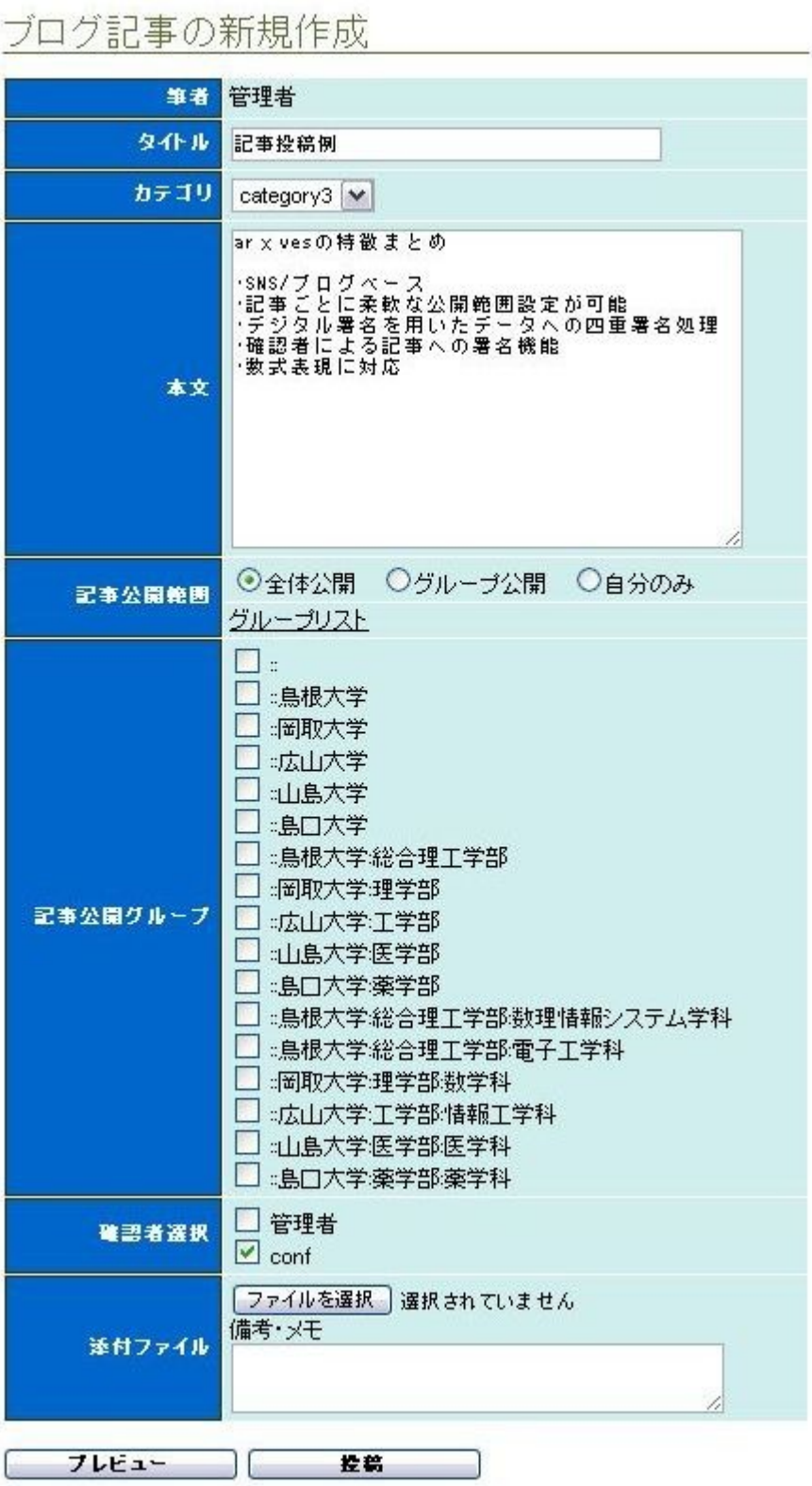
ブログ記事の新規作成
筆者
管理者
タイトル
記事投稿例
カテゴリ
category3
本文
arxivesの特徴まとめ
・SNS/ブログベース
・記事ごとに柔軟な公開範囲設定が可能
・デジタル署名を用いたデータへの四重署名処理
・確認者による記事への署名機能
・数式表現に対応
記事公開範囲
全体公開
グループ公開
自分のみ
グループリスト
記事公開グループ
::
::鳥根大学
::岡取大学
::広山大学
::山島大学
::島口大学
::鳥根大学:総合理工学部
::岡取大学:理学部
::広山大学:工学部
::山島大学:医学部
::島口大学:薬学部
::鳥根大学:総合理工学部:数理情報システム学科
::鳥根大学:総合理工学部:電子工学科
::岡取大学:理学部:数学科
::広山大学:工学部:情報工学科
::山島大学:医学部:医学科
::島口大学:薬学部:薬学科
確認者選択
管理者
conf
添付ファイル
ファイルを選択
選択されていません
備考・メモ
プレビュー
投稿

図9　記事新規作成画面

arχves記事投稿例

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

- -----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

実際のarχvesの記事の投稿例

チェックボックスによる柔軟なグループ指定
公開鍵暗号によるデータの2重署名

数式に対応
$\forall x \in \mathbb{C}\left(\sin^2 x + \cos^2 x = 1\right) + 1$

ClientTimeStamp: 2009-09-07T15:14:49+09:00
- -----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (MingW32)

iEYEARECAAYFAkqkpNkACgkQFfuywYqu77gtgACgnsKLsVLXJE0slWFR63aBtbNt
W4MAoIFcVgCcMFMBJ7fuKXYYE8W6AKAm
=nYec
- -----END PGP SIGNATURE-----

Server TimeStamp. 2009-09-07T15:14:51+09:00
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (MingW32)

iEYEARECAAYFAkqkpNsACgkQNkcjG4aWrL3ErACfTfuY0JyFIjWZwZ6UrheL2wFV
edQAn2t0Pp46ldk5SXiuWF+zSST1v4QF
=ssRz
-----END PGP SIGNATURE-----

公開範囲
:全体公開

図10　2重署名を含む記事全文

2.5.7　アカウントメニュー

「アカウントメニュー」は、「ユーザ」、「ゲスト」がアクセス可能なコンテンツであり、ログインしている利用者自身の情報、投稿データの編集、削除を行うことのできるコンテンツである。ただし、「ゲスト」は自身のアカウントデータ閲覧のみ可能で、編集および削除などは不可能である。メニュー項目について、表4に示す。

表4 メニュー項目

アカウントメニュー	管理者メニュー
アカウント情報の閲覧 アカウント情報の変更 管理グループの変更 ブログタイトルの変更 カテゴリの作成 カテゴリの変更 ブログ記事の閲覧範囲の変更 添付ファイルの削除	ユーザ管理 グループ管理 ブログ記事管理 コメント管理 カテゴリ名管理 ブログ名管理

2.5.8 管理者メニュー

「管理者メニュー」は「管理者」のみがアクセス可能なコンテンツで、システム全体の各データの編集、削除が可能である。

ただし、記事本文の編集は、管理者であっても不可能である。

2.6 記事毎の公開範囲の指定

前述したように、本システムでは、記事単位で公開範囲を指定することが可能である。

公開範囲は「全体公開」、「グループ公開」、「自分のみ」の3種類から指定することができる。

「全体公開」は、システムの認証(ログイン)を受けなくても閲覧可能な公開範囲である。

「自分のみ」は、記事を書いたユーザ自身のみが閲覧可能な公開範囲である。

「グループ公開」は、グループを単位として、公開範囲を指定する。この時、複数のグループを指定することができ、指定されたグループを上位に持つ子グループも、自動的に公開範囲に含む。このような、グループの階層構造を生かした公開範囲制御を行う。

なお、記事の投稿後にも、公開範囲の変更は可能であるが、閲覧範囲は、記事を書いたユーザ、あるいは管理者のみが変更可能である。

3 確認者署名機能の付加

3.1 概要

本システムは前章で示した二重署名処理によって、記事データに対しサーバを第三者とした署名を行っている。しかし、検印サーバによる署名は機械的に付せられた署名であり、記事の内容の確認を行っているわけではない。そこで、本システムに確認者による署名を行う機能を付加することによって、記事の知的財産としての価値を高めるといった観点から改良を行った。以下の節で詳細を示す。

3.2 ラボノートにおける確認者署名

企業、大学、研究機関などで使用されているラボノートには、本人の署名と共に、確認者の署名を行う必要がある。岡崎、隅蔵編による「理系なら知っておきたいラボノートの書き方」[12]によると、ラボノートに要求されることとして、第三者による確認と署名が必要であるとされている。

また、確認者に求められる条件として以下を満たす必要がある。

1. 共同開発者でないこと
2. ラボノートに記載された内容を理解することができること

1.は確認者が共同開発者であった場合、当該研究者と利害が一致するため、ノートの日付、内容などについて、偽造、改竄を行う可能性が他の人をと比較して高いためである。2.はラボノートの内容を確認する人は、機械的に署名をするだけでは不足であり、ノートに記載された実験内容を理解し署名を行う必要があるためである。

3.3 確認者署名処理

3.3.1 記事への四重署名処理

本システムの記事データへの確認署名処理を以下⑥～⑪に示す。確認署名処理は、2.3節で示した二重署名処理終了後の、二重署名処理済み記事データが文書サーバに保管された所から開始する。

⑥記事が投稿されたことを確認者に電子メールで知らせる。
⑦確認者は検印サーバを介し文書サーバへログインし、記事の確認を行う。
⑧二重署名済み文書に対し、確認者署名を行う。
⑨検印サーバでサーバ署名を行い、署名部を保存する。
⑩文書サーバに四重署名済みデータを送る。
⑪文書サーバにデータを保存する。

以下の一連の流れを図式化したものを図10に示す。また、四重署名の行われた実際の文書の例を図11に示す。

また、二重署名処理時と同様、確認者署名とその後のサーバ署名の行われる直前に、データに対して時刻情報を書き込んでいる。

⑦処理時、確認者は検印サーバでIDとパスワードによる確認者の認証を受け、検印サーバにログインを行い、記事データの確認を行う。この際、二重署名済みのデータは文書サーバに保存し

てある為、確認者は検印サーバを介して文書サーバへログインを行い、記事データの取得と確認を行っている。つまり、確認者は検印サーバへログインを行い、検印サーバをproxyとして文書サーバへのアクセスし、文書サーバへログイン、記事データの取得を行っている。実際の検印サーバでの記事本文の表示画面について図12に示す。

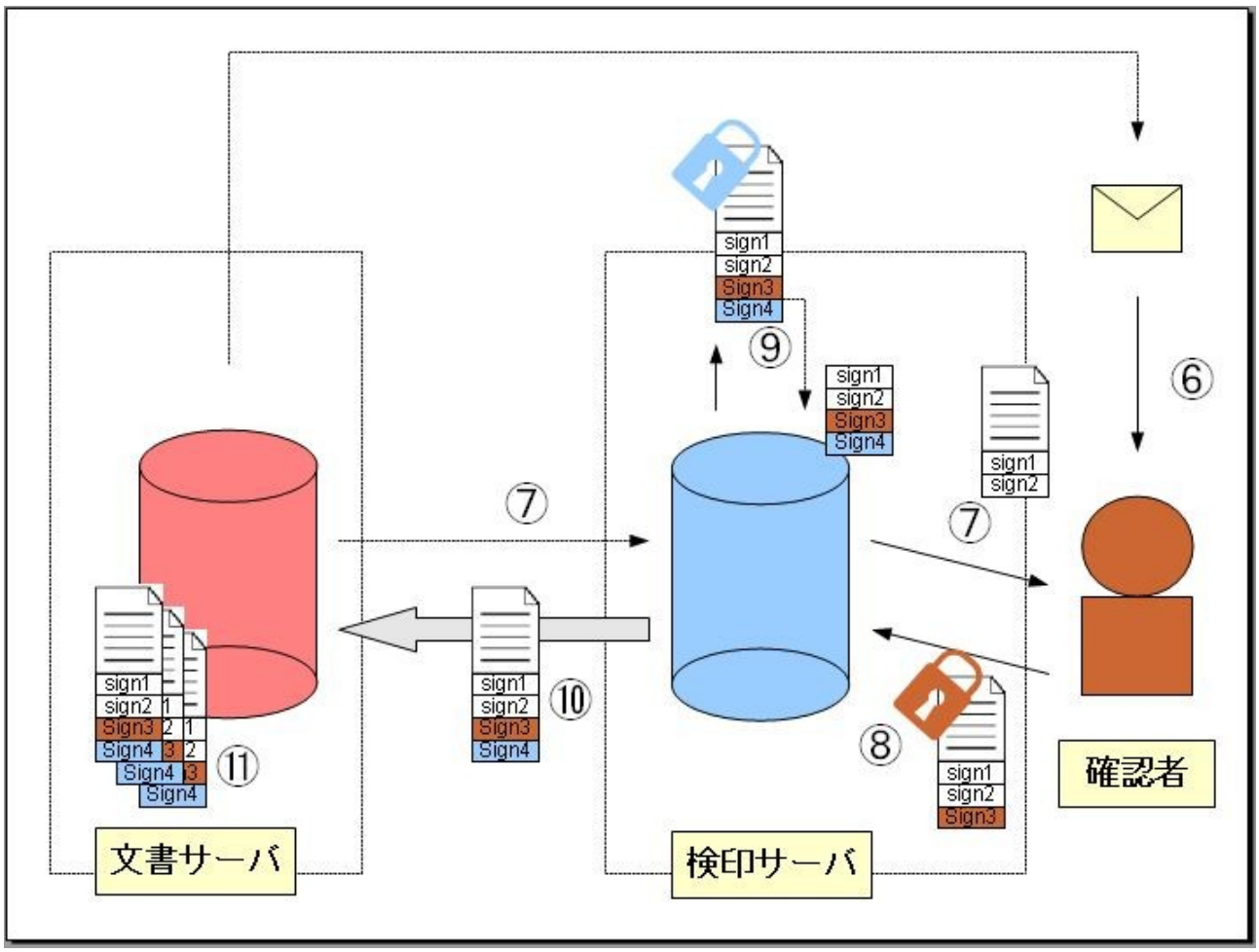

図10　確認署名処理

記事投稿例

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

- -----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

- - -----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

- - - -----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

arXivesの特徴まとめ

・SNS/ブログベース
・記事ごとに柔軟な公開範囲設定が可能
・デジタル署名を用いたデータへの四重署名処理
・確認者による記事への署名機能
・数式表現に対応

ClientTimeStamp: 2010-02-01T13:43:25+09:00
- - - -----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (MingW32)

iEYEARECAAYFAktmWHQACgkQFfuywYqu77j/EgCg5NaHMfFMZGbEyRUiZrBZFZ+H
WvEAni66lq5ArlKAHHnC5Nl4cw9trUt
=TmAy
- - - -----END PGP SIGNATURE-----

Server TimeStamp. 2010-02-01T13:43:26+09:00
- - -----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (MingW32)

iEUEARECAAYFAktmWH4ACgkQNkcjG4aWrL1ezQCYxIKFncogzSZ2gJFrHG1Uj3wW
2wCgIlCDhyV/HRG8LF9BwC6Qhb+cjJs=
=KGAd
- - -----END PGP SIGNATURE-----

ConfirmerTimeStamp: 2010-02-01T13:51:00+09:00
- -----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (MingW32)

iEYEARECAAYFAktmXbQACgkQFfuywYqu77i1OwCgp8xGJZy2E8sHOeDXh3/zJjIQ
VT0AoNGaf6cGJqrSuxoXcxDD4B/jY730
=HDip
- -----END PGP SIGNATURE-----

Server TimeStamp#2. 2010-02-01 13:51:00
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (MingW32)

iEYEARECAAYFAktmXbQACgkQNkcjG4aWrL3pKQCfRb2J0XGRa27f8vbc9A5BNqtM
JLAAnRFGwXcO6HD/QHYSNjZbQrjmqLQq
=cz2i
-----END PGP SIGNATURE-----

図11　四重署名処理済み記事本文

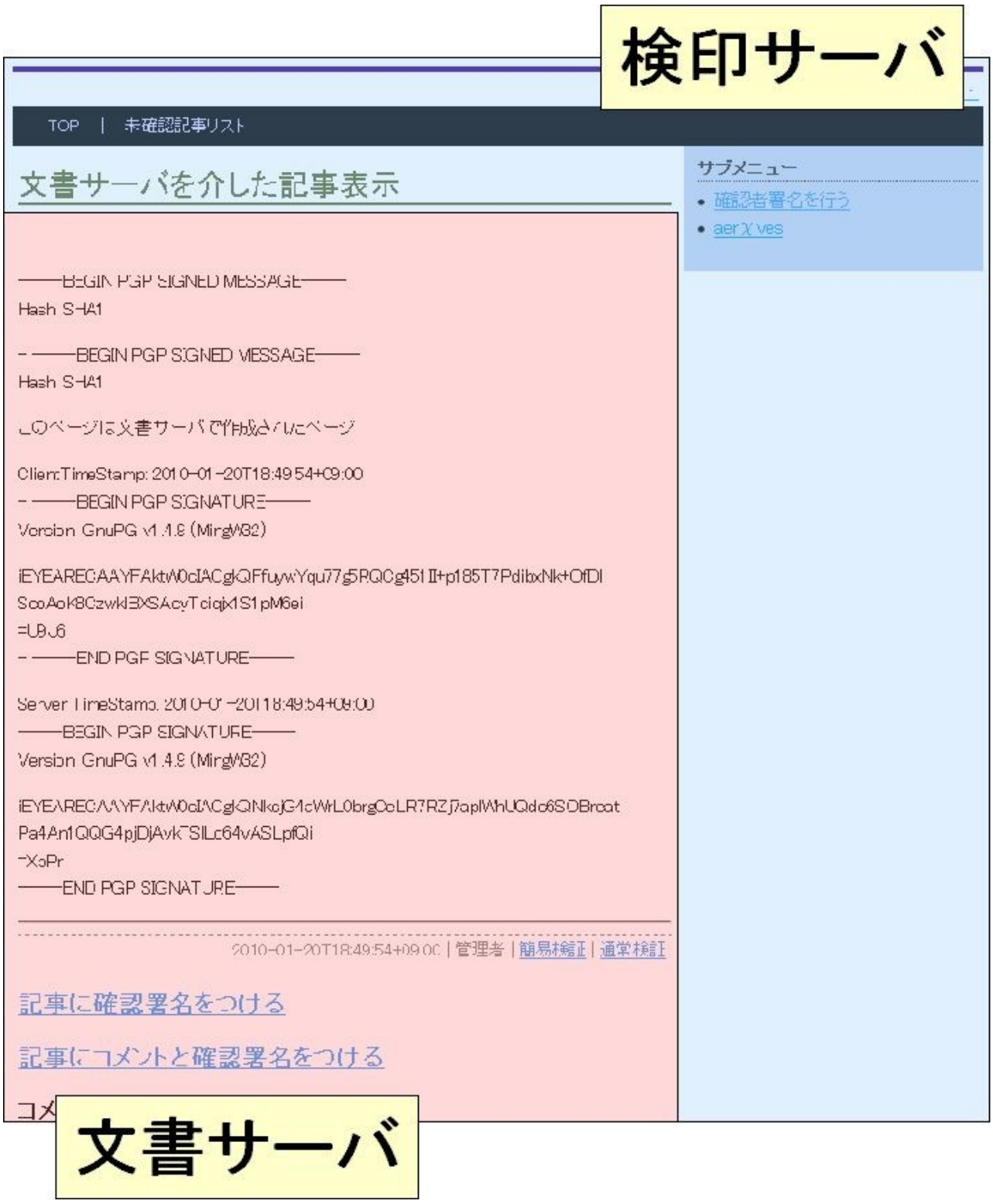

図12　検印サーバを介した記事表示

3.3.2　記事確認依頼メール機能

文書サーバに二重署名済み記事データが保存された時点で、文書サーバは記事投稿時に設定された確認者に対して、記事の確認を依頼する旨の電子メールを送信する。メールの内容は、タイトルに依頼者のユーザ名と投稿時間、本文に記事確認の為のページへアクセスするURLを記載する。実際の確認依頼メールについて図13に示す。

subject: [確認依頼]管理者さんが記事を投稿しました
from: arxves@matsu.shimane-u.ac.jp
reply-to: arxves@matsu.shimane-u.ac.jp
to: [illegible]
date: Fri, 5 Feb 2010 15:29:44 +0900

jump to attachment file

記事タイトル:新規記事作成
記事URL:http://cainan.cis.shimane-u.ac.jp:3000/list/show/36

図13　確認依頼メール

3.3.3　未確認記事一覧表示機能

確認者は記事を確認する際、検印サーバへのログインを行う。このとき、検印サーバ側のトップページのメニューバーのリンクより、未確認記事リストを表示することが可能である。ただし、未確認記事リスト表示を行うためには、検印サーバにログインした後、検印サーバを介して文書サーバへログインを行う必要がある。リストには確認者へ、確認依頼された記事が時系列降順に並ぶようになっており、確認後はリストから削除される。検印サーバのトップページのログイン前とログイン後の画面を、図14、15示す。また、検印サーバログイン後アクセス可能である、文書サーバへのログイン画面について図16に示す。さらに検印サーバを介して文書サーバログインした後に表示可能な、未確認記事リストの例を図17に示す。

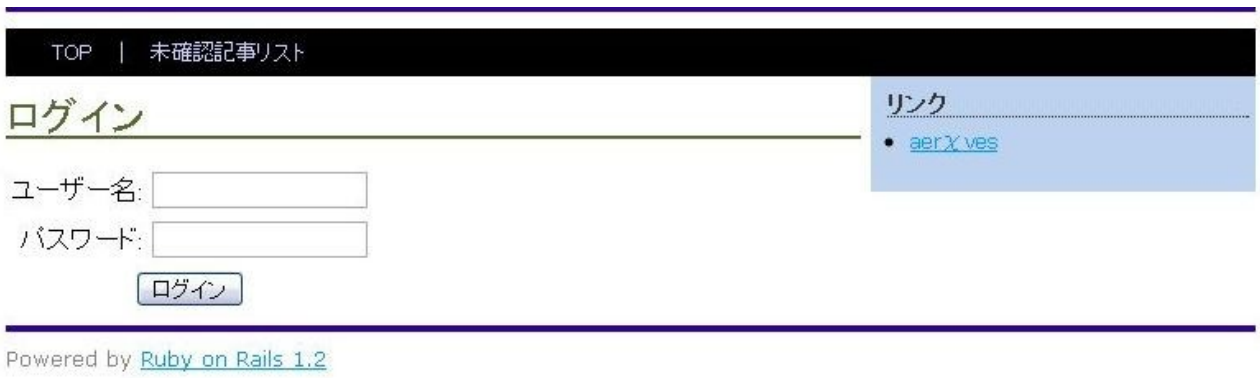

図14　検印サーバトップページ(ログイン前)

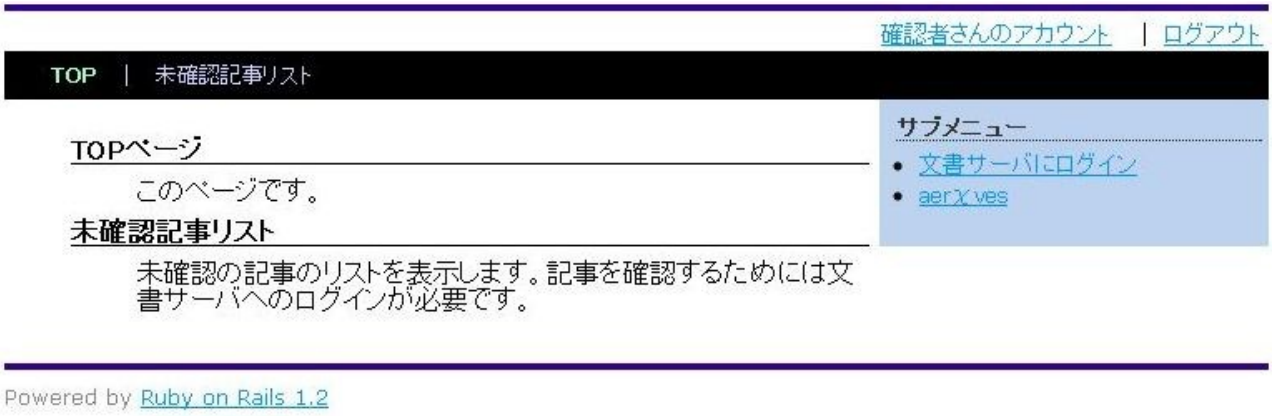

図15　検印サーバトップページ(ログイン後)

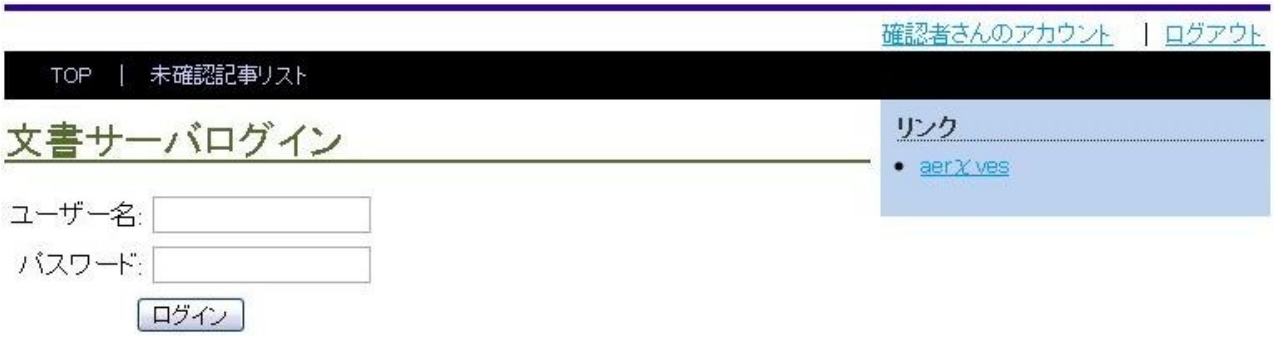

図16　検印サーバを介した文書へのサーバログイン画面

図17　未確認記事リスト

3.3.4　検印サーバでの本文表示および検証

検印サーバで記事を確認する際の記事全文表示画面例を図18に示す。このとき、二章で示した二重署名の検証と同様に検印サーバでの記事確認時に記事データの検証を行うことが可能である。検印サーバでの記事の検証画面の例を図19に示す。

図18 検印サーバでの記事本文表示

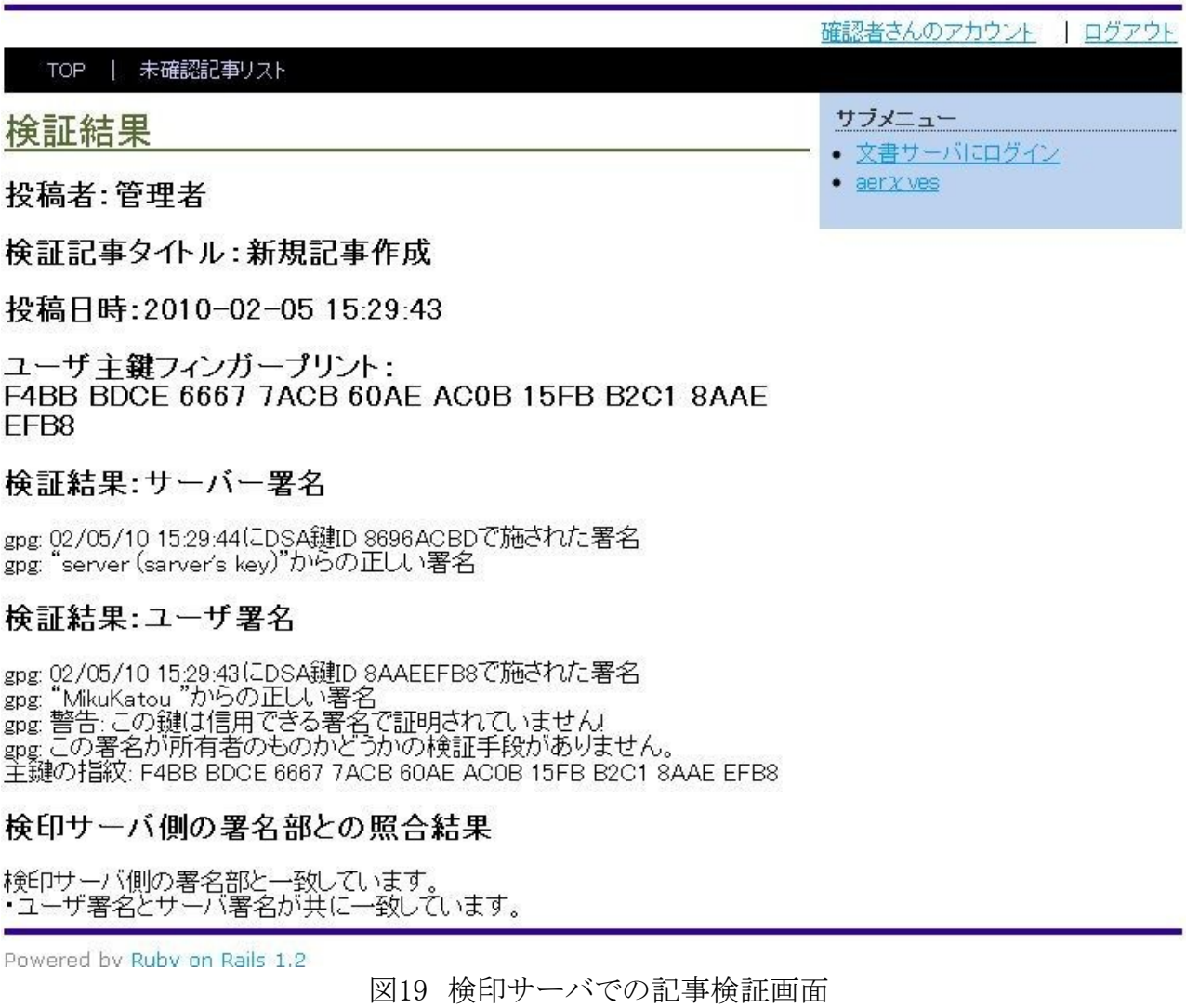

図19 検印サーバでの記事検証画面

3.4 記事及び署名ファイルの4重署名の検証

二章で示した二重署名検証と同様に、確認者署名処理の完了した、四重署名処理済み記事データに対して、検証を行うことが可能である。四重署名の検証を行った画面例を図20に示す。

検証結果

投稿者:管理者

検証記事タイトル:新規記事作成

投稿日時:2010-02-05 15:29:43

ユーザ主鍵フィンガープリント:
F4BB BDCE 6667 7ACB 60AE AC0B 15FB B2C1 8AAE EFB8

検証結果:確認者サーバー署名

gpg: 02/05/10 15:33:14にDSA鍵ID 8696ACBDで施された署名
gpg: "server (sarver's key)"からの正しい署名

検証結果:確認者署名

gpg: 02/05/10 15:33:14にDSA鍵ID 8AAEEFB8で施された署名
gpg: "MikuKatou "からの正しい署名
gpg: 警告: この鍵は信用できる署名で証明されていません!
gpg: この署名が所有者のものかどうかの検証手段がありません。
主鍵の指紋: F4BB BDCE 6667 7ACB 60AE AC0B 15FB B2C1 8AAE EFB8

検証結果:サーバー署名

gpg: 02/05/10 15:29:44にDSA鍵ID 8696ACBDで施された署名
gpg: "server (sarver's key)"からの正しい署名

検証結果:ユーザ署名

gpg: 02/05/10 15:29:43にDSA鍵ID 8AAEEFB8で施された署名
gpg: "MikuKatou "からの正しい署名
gpg: 警告: この鍵は信用できる署名で証明されていません!
gpg: この署名が所有者のものかどうかの検証手段がありません。
主鍵の指紋: F4BB BDCE 6667 7ACB 60AE AC0B 15FB B2C1 8AAE EFB8

検印サーバ側の署名部との照合結果

検印サーバ側の署名部と一致しています。
・全ての署名が一致しています。

図20　四重署名の検証例

4 今後の課題

4.1 セキュリティに関する課題

本システムではいくつかのセキュリティに関する課題が存在する。

正真性の保障について、本システムでは記事投稿時に使用する各ユーザの秘密鍵はDBによる保管を行っている。そのため、ユーザの秘密鍵をICカード及びUSBメモリ等の外部記憶メディアに保管し、記事投稿時に限りそれを参照することにより、正真性の保障を行うことができるであろう。また、本システムにおいて、成りすましが行われる可能性がある。この問題に対しても、ユーザの秘密鍵をICカード、USBメモリ等に保管することにより、成りすましによる記事の投稿を防ぐことができる。

非改竄性の証明の強度向上のため、リンキングやヒステリシス署名、履歴交差などの技術の検討も必要であろう[3、13]。関連して、現在GnuPGに依存したシステムである為、多様な署名方式に対応可能とすることにより、利便性とともに非改竄性の向上が期待できる[14]。また、現在GnuPG使用時、ハッシュ関数sha-1を用いているが、暗号アルゴリズムの2010年問題を考慮し、sha-2等のハッシュ関数に変更する必要がある。

電子署名の長期利用に関して、秘密鍵の危殆化などの問題もあり、評価・検討が必要である[15、16]。同様に一定期間ごとの検証記録を保存することにより、どの時点まで正真性が保持されていたのかを証拠として残すことも今後必要となってくるであろう[17]。

現状では、検印サーバの署名及び確認者署名処理時、サーバ間通信において、記事やデータそのものによる通信を行っている。そのため、情報漏えいの危険性が挙げられる。現在は通信時にSSLを使用することで対処を行っているが、情報漏えい対策としては、十分とは言いがたい。記事やデータのハッシュ値のみによる通信に限る事も含めて、今後検討を要する。また、文書サーバと検印サーバ間の通信時にエラーが起きた場合に、DBへの保存を破棄するなどの対処も必要である。

また、公開鍵の正当性の確保のため、信用の輪(web of trust)も含めたPKIの利用の検討も必要であろう。

4.2 ユーザインタフェースに関する課題

研究支援という観点から、記事中での画像や表・グラフ、化学式への対応が不可欠である。このような問題については、JipsenのASCIIsvg[18]が利用/応用可能であろう。また漢字における異体字や国字に対しての対応も検討していきたい。

その他、現在は記事投稿の際にグループを全て表示し指定することなどに対しての、DHTMLおよびAjax技術などを用いたユーザインタフェースへの改善も今後の課題である。

5 終わりに

本研究では、研究データの改竄を検出可能であり、また研究記録の公開範囲を柔軟に指定できる、研究記録管理・公開・検証システムar χ vesについて、確認者による記事の確認機能の付加による改良を行った。

今後は、四章で取り上げた本システムの課題に対する対応や改良、そして蓄積される記事の知的処理に関して研究を進めたい。

6 謝辞

本研究にあたり、最後まで熱心なご指導を頂きました小林聡准教授に、心より御礼申し上げます。

参考文献

[1]原田　篤史, 西垣　正勝, 曽我　正和, 田窪　昭夫, 「ライトワンス文書管理システム」, 情報処理学会論文誌Vol.44, no.8, pp.2093-2105, 2003
[2]陳　明強, 吉川　正俊, 「時刻認証付きXML文書のデータベースによる管理について」, 電子情報通信学会第16回データ工学ワークシップ（DEWS2005）, 5A-i10, 2005
[3]宇根　正志, 松本　勉, 「可用性および安全性の観点からみた各タイムスタンプ方式間の関係」, 情報処理学会論文誌, vol.43, no.8, pp.2644-2658, 2002
[4]Toshiyuki Masui, Satoru Takabayashi, "Instant Group Communication with QuickML", Proc. ACM Conference on Supporting Group Work(Group '03), pp268-273, 2003
[5]江渡　浩一郎, 高林　哲, 増井 俊之, 「quikWeb:メーリングリストとWikiを統合したコミュニケーション・システム」, 情報処理学会研究報告, 2004-HI-111, pp. 5-11, 2004.
[6] 永田周一, 安村通晃, 「Enzin:情報の公開範囲を手軽に変更できるコミュニケーションツール」, 情報処理学会論文誌Vol.48, no.3, pp.1134-1143, 2007
[7]高井一輝, 河口信夫, 「ACS:多様な人間関係を表現可能なソーシャルネットワーキングシステム」, 情報処理学会論文誌, vol. 48, no. 7, pp. 2328-2339, 2007.
[8]高田　良宏, 笠原　禎也, 毛利　信浩, 松平　拓也, 「多様なアクセス制限に対応した自然科学データベースシステムの開発」, 学術情報処理研究, no.11, pp.50-59, 2007
[9]加藤　未来, 「GnuPG を用いた研究記録管理・公開・検証システム構築」, 島根大学卒業論文, 2008.
[10]加藤　未来, 小林　聡, 「ar χ ves:公開鍵暗号を用いた研究記録管理・公開・検証システム構築の試み」, 学術情報処理研究, No.12, pp.43−51, 2008.
[11]島貫稚華, 「研究記録管理・公開・検証システムar χ vesの数式およびグラフへの対応」, 島根大学卒業論文, 2009.
[12]岡崎、隅蔵編, 「理系なら知っておきたいラボノートの書き方」, 洋土社, 2006
[13]洲崎　誠一, 松本　勉, 「電子署名アリバイ実現機構―ヒステリシス署名と履歴交差」, 情報処理学会論文誌, vol.43, no.8, pp2381-2393, 2002
[14]山田　竜也, 宮地　充子, 双紙　正和, 「オープンネットワークにおける安全な暗号方式の更新に関する考察」, 情報処理学会論文誌　Vol.4, No.8, pp 2102-2109, 2000
[15]宮崎　邦彦, 吉浦　裕, 岩村　充, 松本　勉, 佐々木　良一, 「第三者機関への依存度に基づく長期利用向け電子署名技術評価手法の提案」, 情報処理学会論文, vol.44, no.8, pp1955-1969, 2003
[16]小森　旭, 花岡　悟一郎, 松浦　幹太, 須藤　修, 「署名鍵漏洩問題における電子証拠生成技術について」, 電子情報通信学会「暗号と情報セキュリティシンポジウム」予行集, pp.983−988, 2003
[17] 電子商取引推進協議会, 認証・公証WG, 「電子署名文書長期保存に関するガイドライン」, 2002
[18]Peter Jipsen, ″ASCII MathML″, http://www1.chapman.edu/~jipsen/asciimath.html
[19]黒田　努, 佐藤　和人　共著, 株式会社オイアクス　監修, 「基礎Ruby on Rails」, インプレスジャパン

付録 A　Apatch 設定ファイルの変更

検印サーバ(3000 ポート)から文書サーバ(3500 ポート)のサーバ間通信時、SSL/TLS を使用するためには、以下の設定ファイルを書き換える必要がある。

1.Apatch の SSL 設定ファイル(httpd-ssl.conf)の変更を行う

1)ヴァーチャルホストの設定
<VirtualHost *:2000>
ServerName [プロジェクト名]
DocumentRoot "[アプリケーションの Public ディレクトリ]"
ErrorLog "[エラーログ出力場所]"

<Directory "[プロジェクトパス]\public">
Options FollowSymLinks
AllowOverride all
Order deny,allow
Deny from all
Allow from 127.0.0.1 192.168.0.0
</Directory>

2)フォアードプロキシを Off
ProxyRequests Off

3)静的コンテンツを Apache 側で処理
ProxyPass /stylesheets/ !
ProxyPass /images/ !
ProxyPass /javascripts/ !
ProxyPass /*.html !
ProxyPass /favicon.ico !

4)静的コンテンツのパターンにマッチしなかったリクエストを Mongrel に転送
RequestHeader set X_FORWARDED_PROTO 'https'
ProxyPass /　　http://cainan.cis.shimane-u.ac.jp:3500/
ProxyPassReverse / http://cainan.cis.shimane-u.ac.jp:3500/

付録 B　添付ファイル増強について、基本的なアイディア

現在のシステムは、一つの記事に対して、一つの添付ファイル(material)のみ添付可能である。複数の添付ファイルに対応させる場合の基本的な構想を以下に示す。

複数の添付ファイル owblog(文書サーバ)のコントローラ、blog_kiji_controller にて、5 つの添付ファイルを params[:upload_#{i}](i:0~4)内に所持するよう設計を行った。添付ファイルに署名を行う為には、値が挿入されている params に対し次の 1~3 の処理を行う。

1. それぞれの params を@material に入れる。
2. 各@material を、file_upload 関数に通し、文書サーバでのユーザ署名を行い、署名済み添付ファイルと、添付ファイルの情報を保存する。
3. 各@material を検印サーバへ送信(send_kiji 関数を使用)しサーバ署名を行い文書サーバへ返信する。
4. 文書サーバで、受信した記事データ及び添付ファイルを保存する。

現在のシステムでは 3、において rails(検印サーバ)のコントローラ、sign_controller へ記事データと添付ファイルデータを同時に送信し処理を行い、文書サーバへ返信された署名済み記事データと添付ファイルデータを分割する処理を行っている。添付ファイル数を増やす場合の構想として、次の 2 パターンが考えられる。

- 記事データと同時に添付された全ての添付データを一度に送信する。
- 記事データと添付ファイルを一つずつそれぞれ送信する。

前者の場合、各添付データに対して、@material 及び Material ID（DB）を戻り値に対応できるように送信する必要がある。また、文書サーバへの戻り値の分割作業も改良する必要がある。さらに、一度に大量のデータをやり取りするため、通信エラー等の危険性がある。

後者の場合、各データそれぞれに対して、1 データごとに処理を行う為、Material Id 等を呼び出せるように記録しておく必要はないが、send_kiji 関数及び、sign_controller 等、大幅なコード変更が必要である。具体的なコード改良案について、以下に示す。

- (文書サーバ)blog_kijis_controller の create 関数で各 params を取得、@material に保存。
- 各@material を file_upload 関数に渡し、@material を保存。
- 各@material を send_kiji 関数に渡し、データを検印サーバへ送信する。現状では send_kiji 関数は記事データと添付ファイルを同時に送信するようになっているので、別々に(各データ一つずつ)送信するように改良。
- (検印サーバ)sign_controller の get_kiji 関数で送信されたデータを受け取り、検印を行い、送り返す。検印は kiji モード、material モードによって異なった動作を行うので、文書サーバからデータを送信する際は、モードの情報も付す必要がある。
- (文書サーバ)blog_kijis_controller で署名済みデータを受け取り、データを更新、保存する。
- 各データに対して、以上の動作を繰り返すように設計。
- 確認者署名処理についても同様の改良が必要。

付録 C　サーバ起動方法など

文書サーバ及び検印サーバの起動方法を以下に示す。

文書サーバ起動方法

1. コマンドプロンプトを起動
2. 文書サーバのディレクトリ(C:owblog)へ移動
   コマンド:”cd C:\owblog”　もしくは”cd %rails%”
3. Mongrel を起動
   コマンド:”ruby script/server -p 3500”

検印サーバ起動方法

1. コマンドプロンプトを起動
2. 検印サーバのディレクトリ(C:rails)へ移動
   コマンド:”cd C:\rails”　もしくは”cd %owb%”
3. Mongrel を起動
   コマンド:”ruby script/server [-p 3000]”([]は省略可能)

文書サーバ TOP ページ URL

- http://localhost:3500/main
- http://cainan.cis.shimane-u.ac.jp:3500/main (外部からアクセス可能な URL)

検印サーバ TOP ページ URL

- http://localhost:3000/main
- http://cainan.cis.shimane-u.ac.jp:3000/main (外部からアクセス可能な URL)