研究記録管理・公開・検証システム arχves における確認者署名機能の実現

Realization of Witness Function on arχves: Archive, Publication and Verification System for Research Logs

加藤 康 [1)2)], 島貫 稚華 [1)3)], 藤坂 達也 [1)], 小林 聡 [4)]

Yasushi KATO[1)2)], Chika SHIMANUKI[1)3)], Tatsuya FUJISAKA[1)], Satoshi KOBAYASHI[4)]

skoba@ipc.shimane-u.ac.jp[4)]

島根大学 総合理工学部 [1)]

(現在, (株) プロビズモ)[2)]

(現在, アイシン・コムクルーズ (株))[3)]

島根大学 総合情報処理センター [4)]

Interdisciplinary Faculty of Science and Engneering, Shimane University[1)]

(Probizmo Co., LTD.)[2)]

(AISIN ComCruise Co., LTD.)[3)]

General Information Processing Center, Shimane University[4)]

概要

昨今，研究成果や論文の捏造の問題が社会問題となったことは記憶に新しい．改竄が困難な方法によって研究記録が保存されているならば，指導者あるいは上司による監督や，研究記録あるいは研究実態の検証に関して大いに助けになるであろう．この際，研究記録の正真性と非改竄性の保証をいかに行なうかが課題となる．この問題に対し，我々は電子署名を用いた研究記録管理・公開・検証システム arχves を構築した．しかし，研究記録の証拠能力を高めるためには，記録者の署名のみではなく，確認者 (証人) による署名も必要である．そこで，我々は arχves 上に，同様に電子署名を用いた確認者署名機能を実装した．本システムでは，記事などへの改竄の検証の強度を増すために，確認者の秘密鍵は検印サーバに置き，署名の際には確認者は検印サーバを部分的にプロキシ的に用いて文書サーバと通信し，処理を行なうよう実装した．

キーワード

研究記録, 公開鍵暗号, 電子署名, 確認者, 証人

1 はじめに

昨今，研究成果や論文の捏造の問題が社会問題となったことや，また，本年東京大学において，学位論文に他人の著作物の一部を盗用した箇所が全体の約 4 割にわたって存在するなどの理由により，一旦授与した学位の取り消しがなされたことなどは記憶に新しい．これらの問題の発生を技術的に食い止める事は困難であるが，研

究記録が適切に管理・運用されているならば，指導者あるいは上司による監督や，研究の経緯あるいは研究の実態についての検証に関して大いに助けになるであろう．

しかし，その場合であっても，研究記録は改竄が困難な方法によって管理･運用されていなければならない．この際，研究記録の正真性と非改竄性，および存在証明の保証をいかに行なうかが課題となる．このような課題に対して，我々は電子署名を用いた研究記録管理･公開･検証システム arχves を開発した [1].

しかし，岡崎らは，研究記録の証拠能力を高めるためには，本人による署名の他に確認者 (証人) による署名が必要としている [2]．同書において岡崎らは，確認者の要件として次のようなものを挙げている．

必須要件:

- 発明が理解できる者
- 共同研究者でない者

好ましい要件:

- 将来にわたり，発明について証言可能な者

避けるべき要件:

- 共同発明者の疑念を持たれる共同研究者や上司

「共同研究者でない者」，および「共同発明者の疑念を持たれる共同研究者や上司」については,「ノート記載者と利害関係が強いため，共同発明者の証明は証拠力として不十分」としている．また,「発明が理解できる者」，および「将来にわたり，発明について証言可能な者」については,「漫然と，あるいは，形式的に，署名と日付を記載するのは，証人による署名行為ではない」とし，記録の内容を理解することが必要としている．これは，署名および確認に対する責任の問題が発生するということでもある．

なお，従来の arχves においても，ユーザ署名および検印サーバによるサーバ署名という二重署名を行なっている．しかし，このサーバ署名はサーバによって機械的に行われる署名である．その為，記事の内容を理解した上で行なわれている署名ではなく，署名に対する責任を負うこともできない．

だが，岡崎らも指摘しているように，このような要件を満たす確認者を見付けることは必ずしも容易ではない．しかし，ネットワークを用いれば，地理的に離れた場所にいる人物に確認を依頼することも可能となる．ただし，上記のような要件があるため，実際に面識があり，かつ信用の置ける人物に依頼することとなろう．例えば，指導者自身の出身研究室の教員，あるいは先輩，後輩を第一候補として考えられよう．

そこで，本研究では arχves に対して確認者による確認および署名機能を実現した．

2　システム概要

本研究における arχves の概要図を図 1 に挙げる．arχves は，文書サーバおよび検印サーバからなるシステムである．本システムにおいて，文書サーバと検印サーバは大学程度の規模の組織を隔てて設置されると想定している．

記事の投稿は，ユーザ (3.2 節参照) が文書サーバにアクセスして行なう．記事が投稿された後，確認者は検印サーバにアクセスし，検印サーバを通して文書サーバ上にある記事を確認し，検印サーバ上において署名を行なう．記録の通常の閲覧は文書サーバにアクセスして行なう．ただし，この際にはグループに基づく公開範囲制御が行なわれる．

今回の機能拡張により，arχves の署名は 2 つのフェーズからなる．署名の第 1 フェーズの概要図を図 2 に挙げる．第 1 フェーズは，ユーザが記事を記録，投稿する段階であり，以下の手順からなる．

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なう．
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重に署名済みのデータを送り返す．
5. 文書サーバに二重署名済みデータを保存する．
6. 確認者に記事が投稿された旨の電子メールを送付する．

署名の第 2 フェーズの概要図を図 3 に挙げる．第 2 フェーズは記事に対して確認者が確認を行ない，署名を付ける段階であり，以下の手順からなる．

7. 確認者は検印サーバにログイン後，検印サーバを介し文書サーバへログインし，記事などの確認を行なう．
8. 確認した二重署名済みデータに対し，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．

結果として，記事および添付ファイルには以下の四重

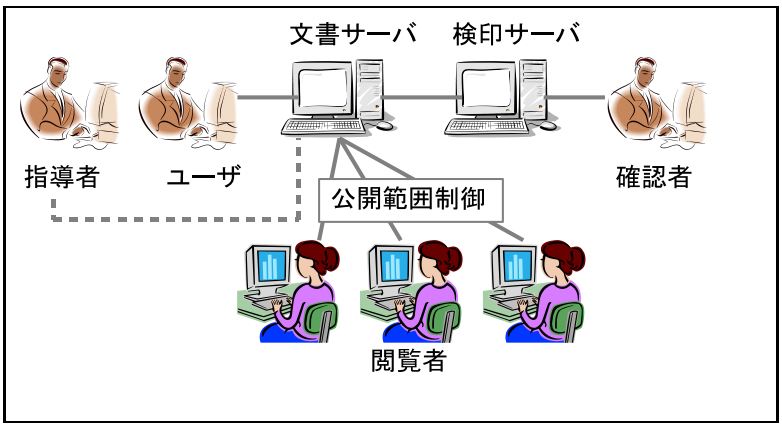

図-1　arχves の概要図

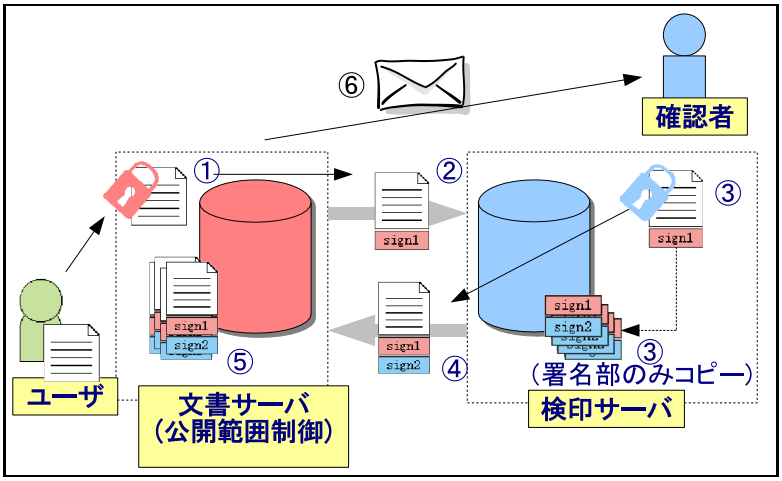

図-2　署名第 1 フェーズ

の署名が付く．

1. ユーザによる署名
2. 検印サーバによるサーバ署名
3. 確認者による署名
4. 検印サーバによるサーバ署名

ここで，ユーザによる署名と確認者による署名は，記事あるいは確認の正真性を担保する意味合いが強く，検印サーバによる両サーバ署名は存在証明の意味合いが強いと言えよう．

3　システムの詳細

3.1　使用したツール等

本システムで使用したツール等を表 1 に示す．

表-1　利用ツール等

開発言語:	Ruby ver. 1.8.6 [*1]
フレームワーク:	Ruby on Rails ver. 1.2.6 [*2]
公開鍵暗号ソフト:	GnuPG ver. 1.4.9 [*3]
ウェブサーバ:	Apache ver. 2.0.6 [*4]
	Mongrel web server ver. 1.1.5
SSL ライブラリ:	OpenSSL ver.0.9.8 [*5]
DBMS:	MySQL ver. 5.0.27 [*6]

Ruby on Rails は，本システムの基幹となる blog 機能の構築を中心に活用した [3]．電子署名の付与および検証については，GnuPG を活用している．Web サーバについては，静的なページは Apache により，動的なページは Apache を経由して Mongrel により生成･送信される．また，文書サーバ ― 検印サーバ間などの通信は https により行なわれている．

3.2　利用者範疇

システム利用者は，文書サーバ上における権限により，「ゲスト」,「ユーザ」,「管理者」の 3 つに分類される．確

[*1] http://www.ruby-lang.org/ja/
[*2] http://www.rubyonrails.com/
[*3] http://www.gnupg.org/
[*4] http://www.apache.org/
[*5] http://www.openssl.org/
[*6] http://www.mysql.com/

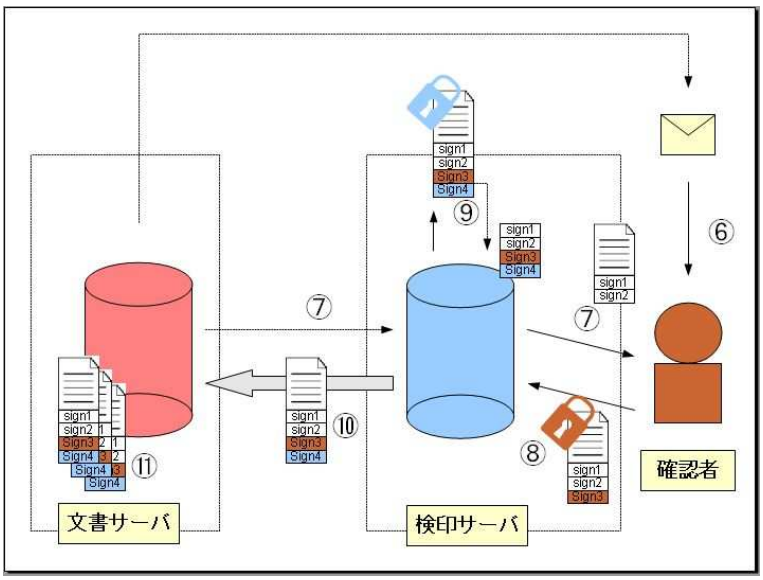

図-3　署名第 2 フェーズ

認者については，文書サーバ上において「ゲスト」でなければならず，またそれとは別個に検印サーバのアカウントも持っていなければならない．

「ゲスト」は，本システムにアクセスした際，認証を通過することにより，当該ゲストが属するグループに対して閲覧が許可されている記事やデータの閲覧が可能となる．

「ユーザ」とは，文書サーバ内に自身の blog を作成し，記事を投稿することができる利用者である．また，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できる．ユーザは，自分が書いた記事の各々について，その記事を閲覧可能なグループを複数個指定できる*7．例外として，公開範囲を指定しないことも可能であり，そのような記事の閲覧については，閲覧者はゲストとしての認証を受ける必要がない．

「管理者」とは，システムの管理者であり，管理者メニューにアクセスできる唯一の利用者である．ただし，あくまで arχves のシステム運用における管理者であり，システム外部の，例えば確認者の選択や依頼などは想定していない．それらは指導者が行なうべきものである。

本システムにおいては，全てのシステム利用者は，1 つ以上のグループに属している．なお，グループに関する情報の編集は，当該グループを登録したユーザ，あるいは管理者のみが可能である．

3.3　署名第 1 フェーズ

署名第 1 フェーズは先の報告 [1] と大きな違いはない．

ユーザは，文書サーバにログイン (図 4) した後に，投稿する記事を書く (図 5).

*7 グループの作成や，閲覧可能なグループの設定は，管理者によっても可能である．

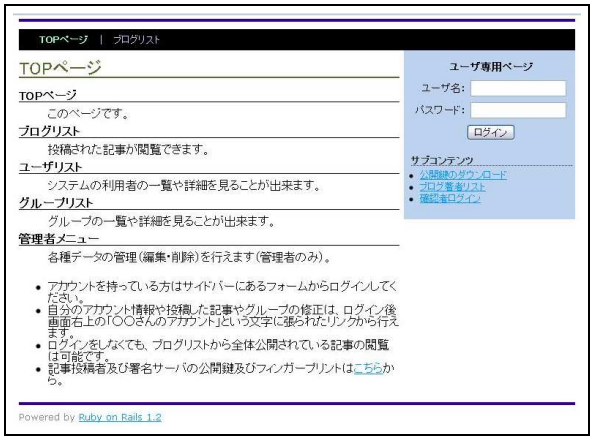

図-4　ログイン画面

記事が投稿されると，文書サーバは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なう．また，その結果を検印サーバに転送する．

検印サーバでは，文書サーバから転送されたデータに対してタイムスタンプを付すとともに，データ全体に対して署名を行なう．この時点で，図 6 のように二重の署名がなされている．二重の署名のうち，最初になされているのがユーザ署名であり，二つめになされているのがサーバ署名である．検印サーバは署名の後，署名部をコピーし，データベースに保管する．その後，検印サーバは二重署名済のデータを文書サーバに返信する．

二重署名済みのデータを受け取った文書サーバは，データをデータベースに保存する．それとともに，その記事を書いたユーザに関連付けられている確認者に，記事が投稿された旨の電子メール (図 7) を送付する．

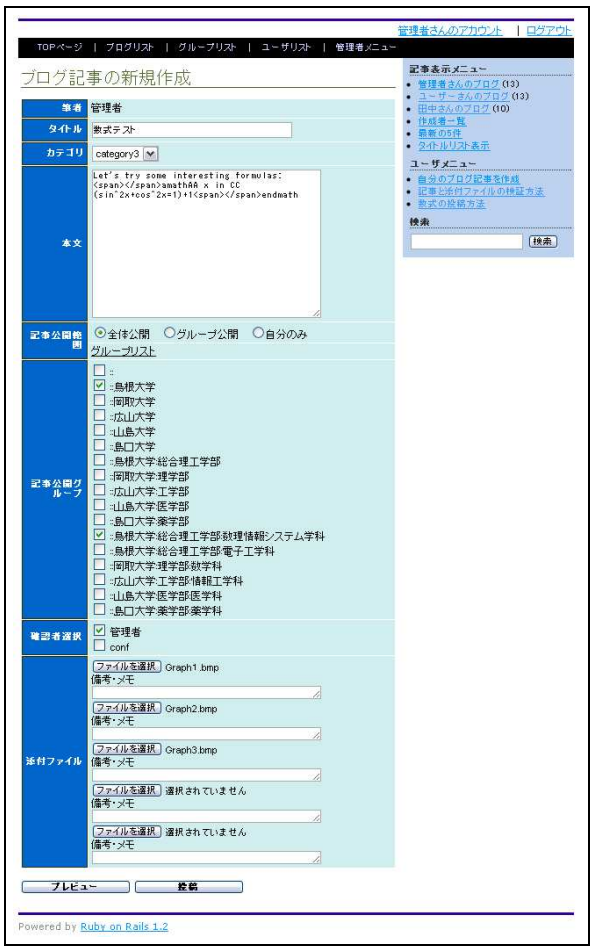

図-5　投稿画面

図-6　署名付き記事

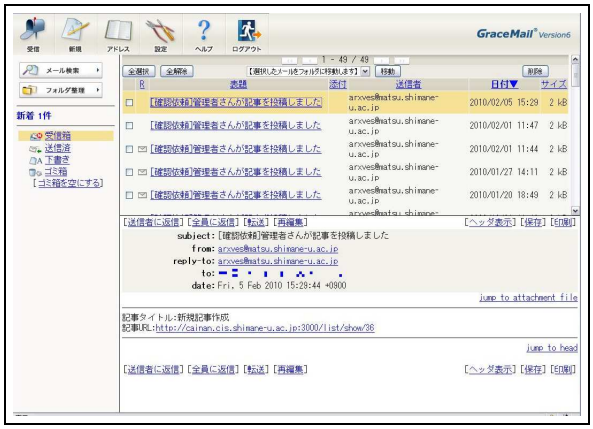

図-7　確認者宛て電子メール

なお，この第 1 フェーズに関連する機能として，先の報告 [1] から，以下の点が改善されている．

まず，記事中において数式を書くことが可能となった．これは AsciiMathML を用いて実現している (図 8)[4]．また，従来，記事投稿時のグループ指定は，テキストボックス内にキーボードから入力していたが，図 5 に見られるように，チェックボックスにより行なうように改善した．記事への添付可能なファイル数も 1 つから 5 つへと増強している．加えて，確認者へのメール送付機能が追加されている．

3.4　署名第 2 フェーズ

署名の第 2 フェーズでは，確認者による署名を行なう．第 2 フェーズでは，原田らによって提案された手法 [7] を拡張し，記事などに更に電子署名を付加する．

確認者は，届いた電子メールから検印サーバにアクセスし，ログインする (図 9)．検印サーバへのログインが完了すると，図 10 のような画面が表示される．このサイドバーから文書サーバにログインを行なう (図 11)．

両サーバへのログインが完了すると，文書サーバ上にある未確認記事の一覧が表示される (図 12)．その中から確認する記事を選ぶと，当該記事が表示される (図 13)．

確認者による記事などの確認時には，確認者は検印サーバを介して文書サーバにアクセスする．その際，検印サーバは部分的にプロキシのように動作し，文書サーバとの通信により受け取ったデータと，検印サーバが独自に提示する部分から，確認者に提示する画面を図 14 のように構成している．なお，この時点で記事の検証を行なうことも可能である (図 15)．記事の確認後，「確認署名をつける」をクリックすることにより，検印サーバ上に置かれている確認者の秘密鍵による署名と，検印サーバの秘密鍵による署名がなされる．この結果，記事などには四重署名がなされる (図 16)．

この後，四重署名部のみを検印サーバのデータベースに保管するとともに，四重署名がなされたデータを文書

図-8　数式への対応

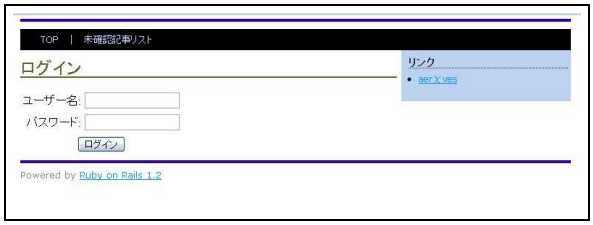

図-9　検印サーバへのログイン画面

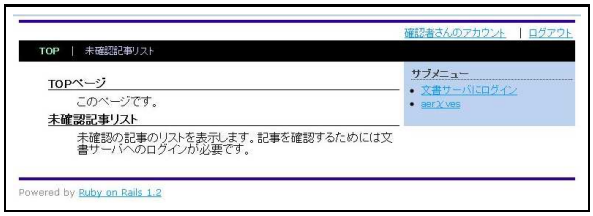

図-10　検印サーバへのログイン後の画面

図-11　文書サーバへのログイン画面

サーバに転送する．文書サーバでは受け取ったデータを保管する．

四重署名に対する検証例を図 17 に挙げる．

4　考察

Blog あるいは CMS を用いた電子研究ノートに関しては轟ら [5] や Ludtke ら [6] による先行研究がある．轟は tDiary を改良することにより実現し，Ludtke は Zope

図-12　未確認記事一覧

図-13　記事確認

を用いて実現しており，いずれも研究記録のグループでの共有が可能となっている．しかし，いずれも記録者による署名および確認者による署名に関する機能は持っていない．この点において arχves の利点がある．

Symyx 社の電子実験ノート[*8] では，電子署名機能を実現している．また，菱化システム社の LabCollector の ELN(電子ノート) モジュール[*9]も電子署名機能を持っており，ノートページの作成者と管理者の 2 者によるバリデーション体制に対応している．しかし，いずれも組織を越えての確認作業は想定していない．これは，私企業において研究記録は知的財産であり，概ね機密に属するものであろうことを考えれば当然のことであろう．arχves では公開範囲制御を行なっているが，研究記録を組織内部の機密とはしない方針である点が異なる．むしろ，arχves は大学などの公的機関における研究記録の管理・公開・検証を主眼としていると言える．

今回実現した確認者による署名に関して，実装上は，

[*8] http://www.symyx.com/jp/productsandservices/software/labnotebooks/index.html

[*9] http://www.rsi.co.jp/kagaku/cs/agilebio/m_eln.html

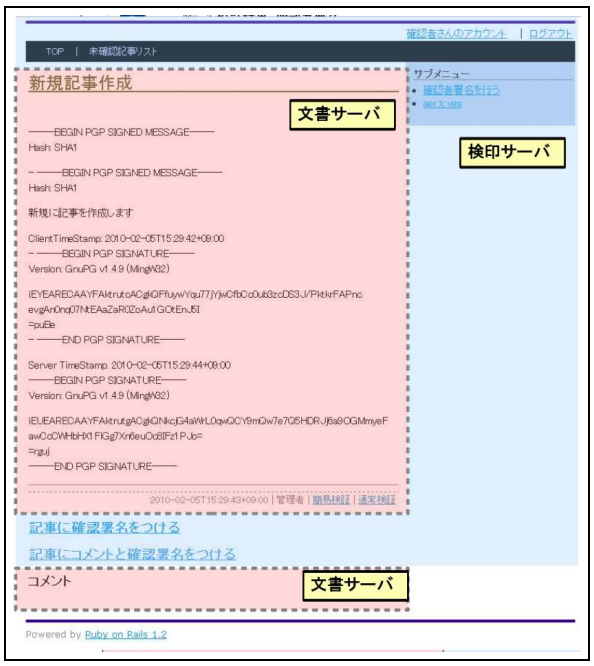

図-14　確認画面構成

図-15　記事検証

ユーザの秘密鍵と確認者の秘密鍵の両方を文書サーバ上に置き，署名を行なう方法が容易である．しかし文書サーバ上にユーザの秘密鍵と確認者の秘密鍵の両方を置く方法では，ユーザあるいは文書サーバの管理者(arχves の管理者とマシンの管理者の両方を含む)により記事などの改竄が可能となる可能性が高まる．そこで確認者の秘密鍵は検印サーバ上に置き，検印サーバ上で署名を行なう方法とした．なお，先の報告 [1] の段階においても，検印サーバは自分自信の秘密鍵を独自に管理しており，記事の改竄を困難なものとしていた．また，二重署名の署名部を検印サーバにも記録し，検証時にはその記録と記事の署名部との比較も行なうことにより，記事の検証の確実さも増していた．

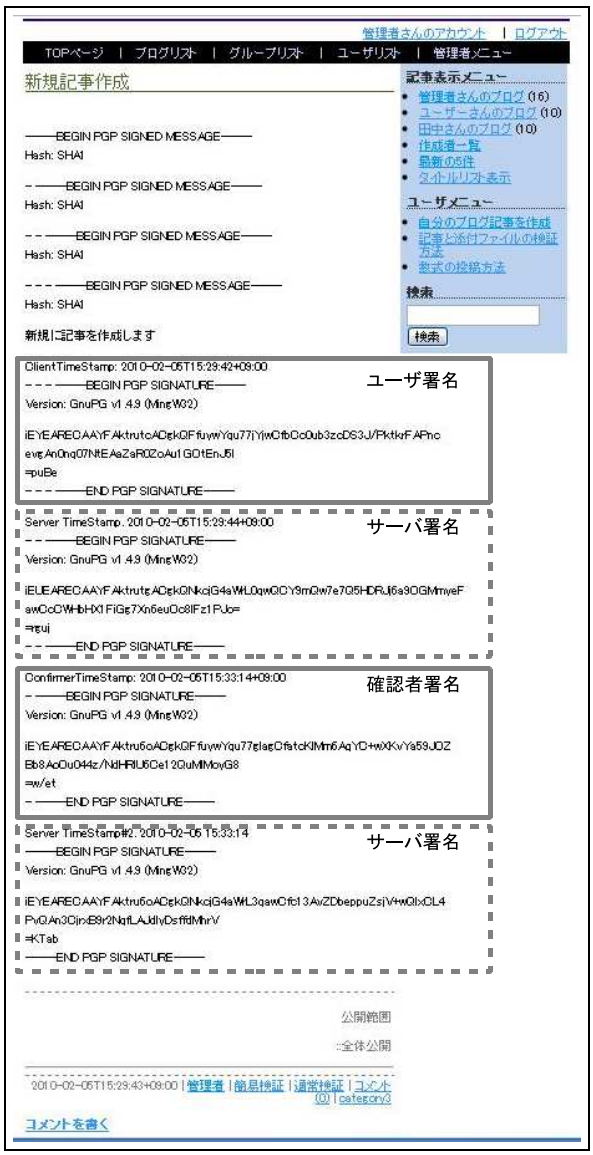

図-16　四重署名済記事

本研究では，arχves 上に確認者による署名機能を実現した．これにより，記録の証拠能力が高まることが期待される．しかし，それはあくまで適切な運用がなされた場合に言えることである．確認者による署名がなされていたとしても，機械的に署名を行なっていたのでは適切な運用がなされているとは言えない．この運用は人間の問題であり，技術面からだけでは解決が難しい．

現状の課題は以下のものが挙げられる．

第 1 に，現状の実装においては，検印サーバを経由して文書サーバにログインする際，検印サーバにおいて確認者の検印サーバに対する ID とパスワード，および文書サーバに対する ID とパスワードを取得可能である．そのため，確認者へのなりすましが可能となっている．この点は早急に対処が必要である．

第 2 に，arχves のユーザ・インタフェースの問題である．記事を公開するグループの選択においては，現状で

図-17　四重署名検証例

はチェックボックスを用いているため，グループ数が多くなった場合には画面上にそれらが羅列されることになり，選択が容易ではなくなる可能性がある．また，添付ファイルに関しては，DB の構造上は一記事に添付可能なファイル数に制限はない．しかし，現状ではユーザ・インタフェースの実装上，上限を設けてある．これらはいずれも，JavaScript + DynamicHTML，あるいは更に Ajax 技術により改善可能である．

第 3 に，記事を投稿するタイミングについてであるが，現状では一日に若干回を想定している．しかし，そのように記事を投稿するタイミングを制限する方向で考えた場合，補助的にではあっても他の媒体による記録が必要となる．他の媒体として，特にノートを用いた場合には，研究記録の物的証拠が残るという強みがある．その反面，他の媒体を経由しての記録では，写し間違いなどが発生する可能性もある．これに対しては，むしろ twitter[*10]や google Buzz[*11]などのように，その時その時の行動をつぶやいておき，記事を書く際には前回から今までのつぶやきの並びとして記事原稿を自動的に構成し，その後にユーザの編集を経て投稿を行なうような機能ないしインタフェースも考えられる[*12]．また，twitter や google Buzz の画面ではフォローしている人のつぶやきも時系列に沿って表示される．個々のつぶやきの管理方法や，記事中での参照方法などについての検討が必要であるが，つぶやきという記録方法やフォローという関係を取り入れることにより，つぶやきあるいは記事間においてある種の履歴交差を自然な形で実現できる可能性がある．

第 4 に，現在，Ruby on Rails は ver. 2.x が標準となっている．そこで，今後研究を継続するためにも，現行の Windows + Ruby on Rails 1 から，Linux + Ruby on Rails 2 あるいは 3 への移植を行なう必要がある．

5　まとめ

従来より開発を進めていた研究記録管理・公開・検証システム arχves に，確認者署名機能を実現した．本システムでは，確認者の秘密鍵は検印サーバに置き，署名の際には確認者は検印サーバを部分的にプロキシ的に用いて文書サーバと通信し，処理を行なうよう実装した．この確認者による署名により，研究記録の知的財産としての価値を高められることが期待される．

今後，4 節に述べた課題について検討，実装を行なう予定である．また，本システム上に蓄積される記録，およびこれまでに別媒体においてなされた記録の自然言語処理および知的処理に関して研究を行なう予定である．

参考文献

[1] 加藤 未来, 小林 聡, “arχves: 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.

[2] 岡崎 康司, 隅蔵 康一, “理系なら知っておきたいラボノートの書き方”, 羊土社, 2007.

[3] 黒田 努, 佐藤 和人, “基礎 Ruby on Rails”, インプレスジャパン, 2007.

[4] Peter Jipsen, “ASCII MathML”, http://www1.chapman.edu/ jipsen/asciimath.xml

[5] S. Todoroki, T. Konishi and S. Inoue “Blog-based research notebook: personal informatics workbench for high-throughput experimentation”, Applied Surface Science Volume 252, Issue 7, pp. 2640-2645, 2006.

[6] Steven J. Ludtke, Laurie Nason, Haili Tu, Liwei Peng, and Wah Chiu, “Object Oriented Database and Electronic Notebook for Transmission Electron Microscopy”, Microsc. Microanal. 9, pp. 556-565, 2003.

[7] 原田 篤史, 西垣 正勝, 曽我 正和, 田窪 昭夫, 中村 逸一, “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.

[*10] https://twitter.com/

[*11] http://buzz.google.com/

[*12] もちろん，このような方法もまた万能ではない．