PIAT を用いた掲示板まとめサイト作成システム

鄭 文龍[†1]　宇田 隆哉[†1]

近年，ブロードバンドの発展に伴ってインターネット上では様々な情報に関して掲示板サイトでの議論が盛んに行われるようになった．これと同時に，掲示板サイトに書きこまれた内容を分かりやすくまとめて掲載する掲示板まとめサイトも増加してきた．しかし，掲示板まとめサイトでは内容を転載する際に同じような意味の書き込みを削除しているため，元の掲示板サイトに掲載されている内容と比較して何らかの改ざんが行われていないかどうかの検証が難しいという問題点がある．また，まとめサイト内では削除された書き込みに含まれていたキーワードを用いた検索も行えないため，この二つの問題点を解決するために PIAT 署名とブルームフィルタを用いた掲示板まとめサイト作成システムの提案を行う．

System for Article Summary on Message Boards with PIAT

WENLONG ZHENG[†1]　RYUYA UDA[†1]

The discussion about various information become increasingly active on the internet message board as broadband developed. As a result , website that publish content of the message board had increased rapidly. However, site that publish the contents are most likely deleting some contents of the original message. Moreover, it is very difficult to verify falsification between original message board and summarized websites. In addition, retrieval of articles using keyword in article summary cannot be done in site for summarizing article. This paper proposes system for summarizing contents on message board with PIAT and Bloomfilter to solve these two problems.

[†1] 東京工科大学大学院バイオ・情報メディア研究科
Graduate School of Bionics, Computer and Media Sciences, Tokyo University of Technology

1. はじめに

近年ブロードバンド回線の普及に伴って様々な情報コンテンツの配信が盛んに行われるようになった．また，これらの発信された情報についてネット上の掲示板で意見交換がされることが増加してきた．

特に日本有数の巨大掲示板となった2ちゃんねるでは毎日500件にも登る数のニュースや情報関連のスレッドがたてられており，さらにそれらのスレッドの内容を分かりやすくまとめて掲載している掲示板まとめサイトも増加してきた [1)]．

これらのまとめサイトでは元のスレッド上にある同じような内容の書き込みや，特定の書き込みに対する反応などを削除して全体の長さを縮め，スレッドの流れが分かりやすくなるようにしている．また，2 ちゃんねるの専用ブラウザなどを知らない，あるいは知っていても詳しい知識がないために扱えないというようなユーザでも簡単にまとめサイトを閲覧することが出来るため，現在利用者が急増しており，開設 1 年足らずで 1000 万アクセスを達成したまとめサイトも存在している [2)]．

しかしながら，掲示板まとめサイトではスレッドの内容を記載する際に元々あった書き込みを削除してしまうため，まとめサイト側に記載されている内容が何らかの改ざんを受けていないかどうかを判断することが難しく，また削除された書き込みに含まれていたキーワードを用いてまとめサイト内で検索を行うことが出来ないといった問題点が存在する．

2. 関連技術

2.1 墨塗り署名

電子文書の安全性を高める技術として電子署名技術があり，電子文書に対するいかなる改変をも検知できるように設計がなされており，不正者による改ざんから電子文書を保護する方法としては非常に有効である．しかし，この強固なセキュリティは電子文書を活用する際に障害となることも多く，例えば行政機関における情報公開の際に個人のプライバシ情報などを削除した場合にも不正者によって改ざんを受けたと見做されてしまうといった問題がある．

上記の問題を解決するために元の電子文書の一部を秘匿したあとであっても，秘匿部分以外の範囲における同一性の検証が可能な電子署名技術として墨塗り署名が存在する [3)]．墨塗り署名におけるエンティティは署名者(signaer)，墨塗り者(sanitizer)，検証者(verifier)の三者で，署名者は既存の電子文書に対する署名を生成，墨塗り者はその電子文書から墨塗り文章を作成，最後に検証者は署名検証により墨塗り文章で開示されている部分についての完全性を確認する．以下に墨塗り署名の流れについて解説する．この時署名者と墨塗り者はあらかじめ公開鍵・秘密鍵ペアを所持しているもの

とする．

2.1.1 署名生成手順

① 署名者は元の電子文書を n 個の部分文章に分ける．
② n 個の部分文章それぞれに対して識別子となる乱数を生成し，部分文章と対応する乱数を結合させたデータを生成する．
③ 各乱数付きの部分文章データのハッシュ値を算出し，**図 1** のように算出された n 個のハッシュ値を結合したハッシュ値集合 h に対し，署名者の秘密鍵 Sk を用いて署名を生成する．
④ n 個の乱数付き部分文章からなるデータを文章 M とし，この文章 M と③で生成した署名を墨塗り者に渡す．
⑤

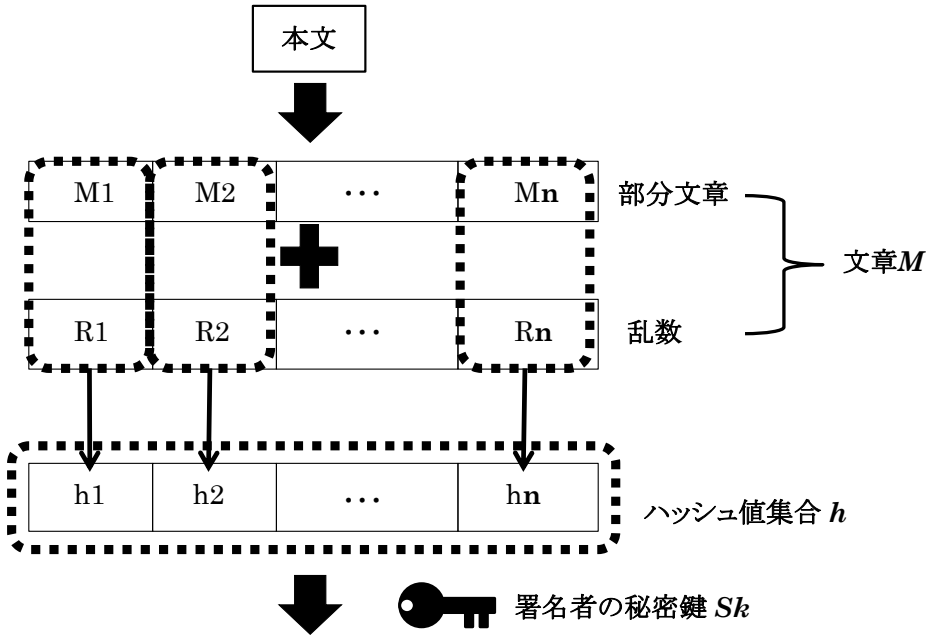

図 1　署名生成処理
Fig.1　Process of Signature generation

2.1.2 墨塗り手順

① 開示対象である文章 M の中から開示しない部分文章を選択し，墨塗り処理を行う．
② ①で墨塗り処理がなされた文章 M と署名者の署名 ρ を**図 2** のように開示文章として検証者に渡す．

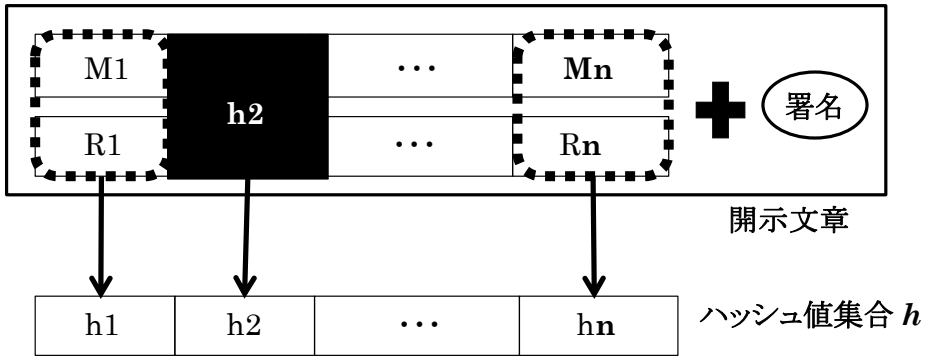

図 2　開示文章
Fig.2　Construction of document for disclosing

2.1.3 検証手順

① 検証者は署名者の公開鍵 Pk を用いて開示文章の署名部分を復号し，ハッシュ値集合 h を算出する．
② 開示文章のうち，墨塗り処理が施された部分文章以外の乱数付き部分文章のハッシュ値を算出する．
③ ②で算出されたハッシュ値と，墨塗り処理が施された部分文章に記載されているハッシュ値を結合させ，①で算出したハッシュ値集合 h と比較検証を行う．

2.2　PIAT 署名

PIAT 署名とは 2004 年に武仲等が提案した墨塗り署名方式 [4]であり，従来の墨塗り署名が持っていた開示部分の完全性と非開示部分の秘匿性を損なわずに，複数の墨塗り者が存在していた場合に墨塗り者とその墨塗り箇所の特定が可能，不正な墨塗りが行われた場合にそれを行った墨塗り者とその箇所の特定が可能，さらには墨塗りが施された部分文章の内容を墨塗り者が自由に変更可能であるという付加要素を備えている．以下に PIAT 署名の手順を示す．また，2.1 と同様に署名者とすべての墨塗り者はあらかじめ公開鍵・秘密鍵ペアを所持しているものとする．

2.2.1 署名生成手順

署名者は 2.1.1 と同じ手順で大元の n 個の部分文章を持つ電子文書から文章 M とハッシュ値集合 $h^{(0)}$，そして自身の署名 $\rho^{(0)}$を生成し，これらを墨塗り者に渡す．

2.2.2 墨塗り手順

PIAT 署名では複数の墨塗り者が想定されており，今回は j 人の墨塗り者がいるものとして，手順の解説を行う．

① 文章 M と 2.2.1 で生成されたハッシュ値集合 $h^{(0)}$と署名 $\rho^{(0)}$のﾍﾟｱ($h^{(0)}$，$\rho^{(0)}$)を受け取った最初の墨塗り者は，墨塗り処理を施す部分文章を新たなデータに置き換えて文章 $M^{(1)}$を生成する．

② ①で生成した $M^{(1)}$から)新たなハッシュ値集合 $h^{(1)}$を生成する．

③ 墨塗り者は自身の秘密鍵を用いて $h^{(1)}$から署名 $\rho^{(1)}$を生成し，$M^{(1)}$とともに次の墨塗り者に渡す．

④ J 番目の墨塗り者は一番目の墨塗り者と同じ手順で非開示の部分を定め，文章 $M^{(j)}$ とハッシュ値集合 $h^{(j)}$ ，そして自身の署名 $\rho^{(j)}$を出力する．

2.2.3 検証手順

① 検証者は最後の墨塗り者が出力した文章 $M^{(j)}$ と j+1 個のハッシュ値集合・署名のペア($h^{(0)}$， $\rho^{(0)}$),…, ($h^{(j)}$， $\rho^{(j)}$)を受け取る．

② 文章 $M^{(j)}$からハッシュ値 $h_i(i=1,\ldots,n)$を求め，これが $h^{(j)}$と等しくない場合は invalid を出力して終了する．

③ 各ハッシュ値集合と署名のペアを対応する公開鍵を用いて検証を行う．検証に失敗した場合には invalid を出力して終了する．

④ 2 組のハッシュ値集合 $h^{(0)}$ と $h^{(j)}$を比較して，墨塗り処理が施された部分文章の場所を特定する．

⑤ j+1 個のハッシュ値集合 $h^{(0)}$,…,$h^{(j)}$を用いて，墨塗りされた i 番目の部分文章の墨塗り者を特定する．この時，$h_i^{(0)}=\cdots=h_i^{(k-1)}\neq h_i^{(k)}=\cdots=h_i^{(j)}$となる $k(k=0,\ldots,n)$を求め，k 番目の墨塗り者が i 番目の部分文章の墨塗りを行ったと判定する．

⑥ すべての墨塗りされた部分文章について⑤の作業を行い、墨塗りを行った墨塗り者を全員特定できた場合には valid を，そうでない場合には invalid を出力して終了する．

2.2.4 墨塗り者と不正墨塗り者の特定

すべての墨塗り者は秘密鍵を用いて署名を出力しているため，署名検証を通じて誰が墨塗り者であるのかを特定することができる．また，同時にハッシュ値集合も出力しているため，これらの情報の整合性の検証を通じて不正墨塗りの箇所とそれを行った墨塗り者を特定することが出来る．

2.2.5 PIAT 署名の問題点

PIAT 署名は従来の墨塗り署名と比べて電子文章の内容を改変しても署名検証が可能であるという特徴がある．しかしながら，大元の電子文章が Web ページの一部であり，墨塗り文章が他の Web ページで公開されている場合に，検証者がどこからどこまでが墨塗り文章であるのかをはっきりと特定することが出来ず，署名検証を行えないという問題点がある．

2.3 ブルームフィルタを用いた検索手法

近年クラウドコンピューティングサービスの一つである Database as a Service(DAS)が注目を集めている．DAS の利用者はデータベースを設置・管理するための労力をかけること無く高性能なデータベース昨日を利用することが可能であるが，サービスの管理者がデータ所有者と異なるため，データ所有者が管理者に対してデータベース内の情報を守る必要性が生じている．そこで渡辺等は 2008 年にブルームフィルタを用いたプライバシ保護検索手法を提案している [5]．以下にこの手法の概要を示す．

2.3.1 データベースの変換

データベースには保存したい内容ではなく，その内容を暗号化した情報と検索用のブルームフィルタを保存する．**図 3** にその例を示す．**図 3** では本来ならば ID と date(日付)と content(内容)のように保存される情報が変換され，サーバにはタプル t 全体を暗号化した etaple と検索用の索引である bfindex のみがアップロードされる．これによって第三者にはこのデータベースに保存されている情報を得ることが出来なくなる．

2.3.2 ブルームフィルタの作成

図 4 に検索用のブルームフィルタ(bfindex)を生成する際の流れを示す．まずタプル t から語の集合である Wt={w0,…,wn}を生成する．各語は属性名と属性値からなる．属性値が文字列である場合には，部分一致用に単語毎や n-gram に従って分割し，属性名と合わせて語とする．これらの語に対して HMAC(鍵付きハッシュ関数によるメッセージ認証関数)を適用し，その値に基づいてブルームフィルタにビットを立てる．

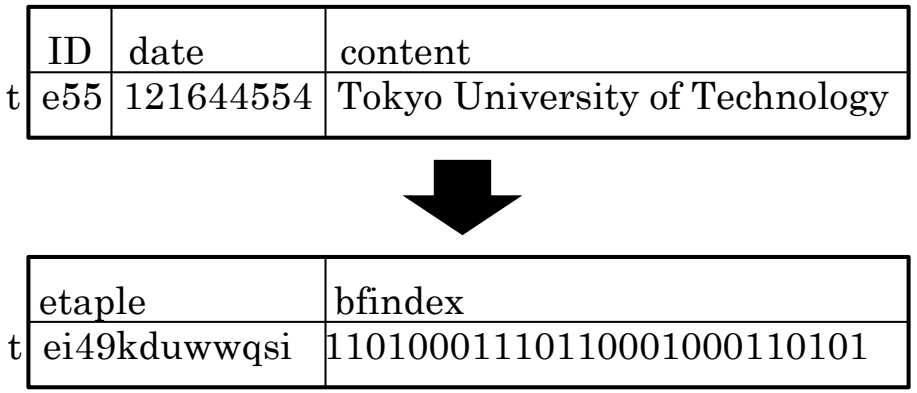

図 3　データベースの変換例

Fig.3　Example of index transformation on database

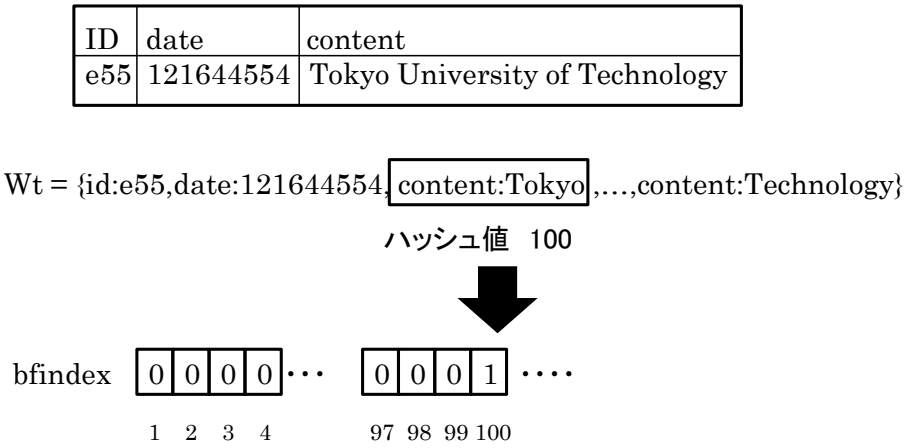

図 4　タプルから索引を生成する流れ

Fig.4　Index generation on tuple

3. 提　　案

本章では PIAT 署名とブルームフィルタを用いて転載元となった掲示板とまとめサイトの間で署名検証が可能であり，かつまとめサイト内で削除された書き込みに含まれていたキーワードを用いた検索も可能であるような掲示板まとめサイト作成システムの提案を行う．

3.1 概　　要

本手法ではまとめサイトの製作者とは関係の無いボランティアにより設置された署名作成及び検証を行う検証サーバが大元の掲示板とまとめサイトの双方にアクセスして Web ページの内容を取得して署名の検証を行い，まとめサイトを閲覧しようとするユーザはブラウザのアドオン機能を用いてこの検証サーバにアクセスして検証結果を得るものとする．また，まとめサイト作成者は自身のサイト内での検索は後に述べるようなブルームフィルタを用いるように実装するものとする．

提案手法の概要を**図 5** に示す．

① まとめサイト製作者は 2 ちゃんねるから自身のサイトに転載したいと思ったスレッドの HTML コードを取得し，これを用いてブルームフィルタを生成する．その後，必要ないと考えた書き込みを削除してから残りの部分を自身のサイト内にて掲載する．また，この時に掲載したのと同じページ内に元のスレッドの URL をサイトを閲覧するユーザに対して明示的に記述する．

② まとめサイトを閲覧しようとするユーザはサイト内に記述されている元のスレッドの URL とサイト自体の URL をブラウザのアドオン機能に入力する．

③ ブラウザのアドオンは 2 つの URL を検証サーバに送信し，検証の依頼を行う．

④ 検証サーバは受け取った URL 先の HTML コードを掲示板とまとめサイト双方から取得し，それぞれ PIAT 技術を用いて署名を生成して検証を行う．

⑤ 検証サーバは検証に成功した場合は valid と検証の結果判明したまとめサイト内で掲載されている書き込みの番号を，失敗した場合は invalid をまとめサイトを閲覧しているユーザのブラウザのアドオンに対して送信する．

⑥ ユーザのブラウザのアドオンは検証サーバから valid を受け取った場合は検証成功とまとめサイト内に掲載されている書き込みの番号を，invalid を受け取った場合は検証失敗をユーザに対してポップアップにて知らせる．

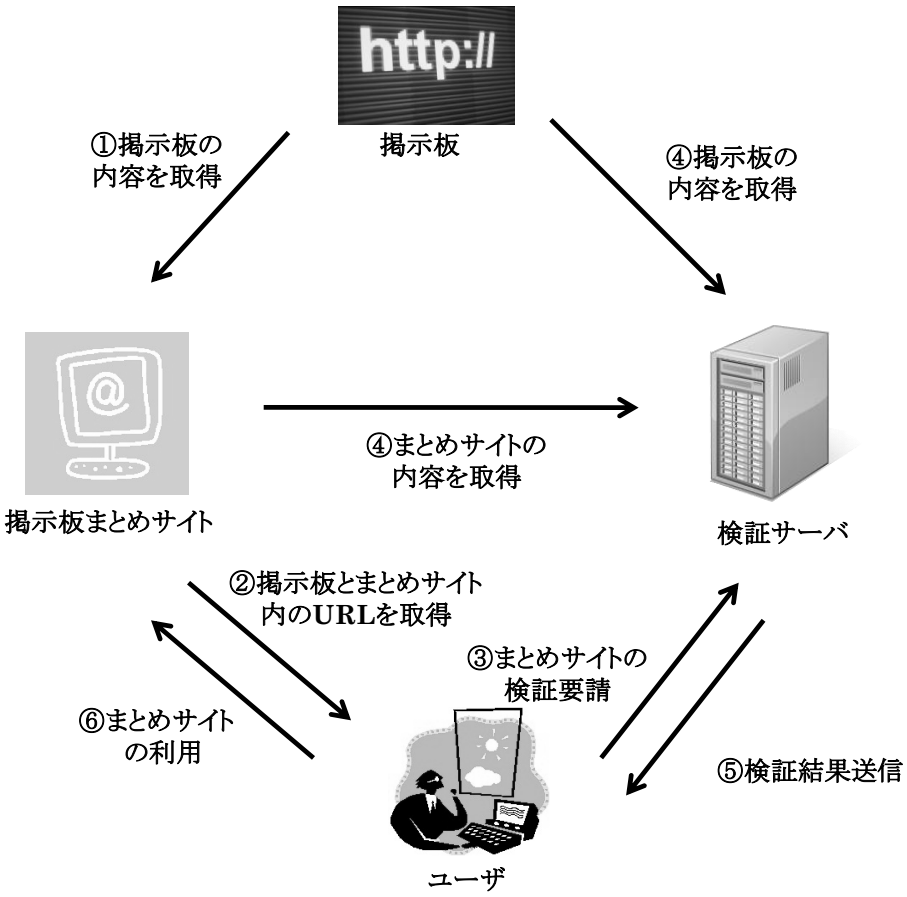

図 5　提案手法の概要
Fig.5　Proposed method

3.2　まとめサイト

3.2.1 ブルームフィルタの生成

まとめサイト製作者は自身がまとめたいと考えた 2 ちゃんねるのスレッドの HTML コードを取得し，それを用いて ブルームフィルタを生成する．この時，ブルームフィルタの長さを 2^{16}=65536bit とする．この理由は検索を行う際には「固有名詞」,「組織名」,「人名」を用いるのが一般的であり，2 ちゃんねるのスレッドの書き込み数は 1000 が上限であるため，ひとつのスレッド内に存在する上に挙げたような名詞の数は 2^{16} 個以下であると推測されるからである．

2 ちゃんねるのスレッド内の各書き込みは HTML コード上では**図 6** のような構造となっている．また，その構成要素について**表 1** に詳細を示す．

まとめサイト製作者は *writing* の内容から形態素解析を用いて「固有名詞」,「組織名」,「人名」の三つの名詞を抽出し，これらを任意のハッシュ関数 *H()*を用いてハッシュ値を算出して，その先頭の 16bit を用いて 2.3.2 で説明した方法でブルームフィルタを生成する．

<dt>*id**name**date*<dd>*writing*
writing

図 6　書き込みの構造
Fig.6　Construction of article on message board

表 1　書き込みの構成要素
Table.1　Elements of article on message board

要素名	備考
id	書き込みの番号
name	書き込みを行った人物が自由に設定できる自身を表す名前
date	書き込みが行われた日時
writing	書き込みの一行ごとの内容

3.2.2 ブルームフィルタを用いた検索

まとめサイト製作者は自身のサイト内検索機能を以下のような手順でブルームフィルタを用いて行うように実装するものとする．

① 検索フォームにキーワードが入力された場合，ブルームフィルタ生成時に使用したハッシュ関数 *H()*を再び用いてキーワードのハッシュ値を算出する．

② ①で算出されたハッシュ値の先頭 16bit の値 *x* を用いて，自身のサイト内に存在するブルームフィルタの中から *x* 番目のビットが 1 となっているブルームフィル

タを探し，このブルームフィルタの生成元の Web ページを検索結果として出力する．

3.2.3 スレッドを転載する際の HTML コード

まとめサイト製作者が 2 ちゃんねるからスレッドを転載してくる際には，HTML コード内に**図 7** のようにコメントアウト文を用いてスレッドの内容の掲載開始場所と終了場所，および削除を行った書き込みの番号と場所について検証サーバにたいして明示的に示す．「thread start」はこの次の行からスレッドの内容の掲載を開始するという合図であり，「thread finish」は終了の合図である．また，「delete writing start」は ここから削除が始まったことを示しており，そのすぐ後の“”の中に最初に削除された書き込みの番号や書き込んだ人物の名前，そして書き込み時間を記述する．また，「delete writing finish」はここで削除が終了していることを示しており，そのすぐ後の“”の中には同様にして最後に削除された書き込みの内容を記述する．

図 7 では例として 100 番目から 105 番目の書き込みを削除した場合の書式について記述している．コメントアウト文の「delete writing start」と「delete writing finish」の間にある HTML コードについて検証サーバは無視するため，まとめサイト製作者が広告などを入れることが可能である．

```
<!--thread start-->
      ⋮
<!--delete writing start“100：名前：2011/05/24(火) 23:30:01.91”-->

                  広告等

<!--delete writing finish “105：名前：2011/05/24(火) 23:50:31.01”-->
      ⋮
<!--thread finish-->
```

図 7　削除した書き込み部分の HTML コード
Fig.7　HTML explanation in deleted article

3.3　検証サーバ

3.3.1 署名の作成

以下に検証サーバが署名を生成する際の手順を示す．

① 検証サーバはユーザのブラウザのアドオンから送信されたきた 2 つの URL のうち，2 ちゃんねる側の URL から署名を生成するために HTML コードを取得する．

② 2 ちゃんねるのスレッドの HTML コードの構造は**図 6** の用になっていることから，**図 8** のように 1 行ごとの書き込みをひとつの部分文章とする．また，部分文章の識別子として各書き込みの *id* と *name*，*date* を連結したものを用いる．よって同じ番号の書き込み内にある部分文章はすべて同じ識別子を用いることになる．

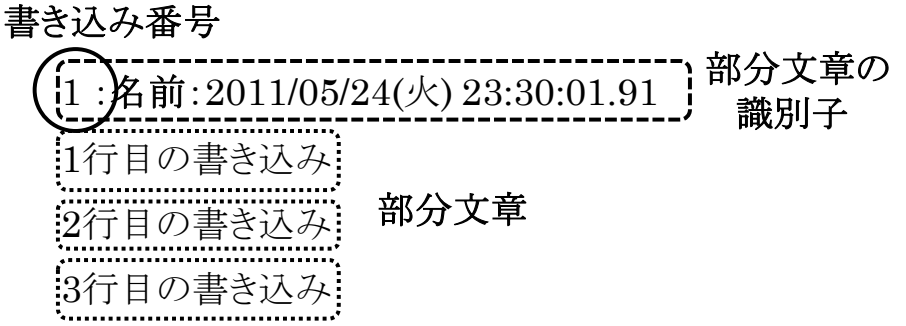

図 8　部分文章とその識別子
Fig.8　Construction of identifier and article

③ 部分文章と識別子を連結させたものから任意のハッシュ関数 *H()*を用いてハッシュ値を算出する．このハッシュ関数 *H()*は 3.2.1 で用いられたものと同一のものである必要はない．

④ すべての部分文章と識別子のハッシュ値を算出したら，これらを連結して検証サーバは任意の安全な(適応的選択文章攻撃に対して存在的偽造不可能な)署名方式の関数 $\mathrm{Sign}_{sk}()$と検証サーバ自体の秘密鍵を用いて署名を生成する．

⑤ 検証サーバは一度生成した署名を保存しておき，複数のユーザが同じまとめサイトの検証を依頼して来ることに備える．**図 9** に署名を保存するときのデータ構造を示す．また，その構成要素について**表 2** に詳細を示す．

id	url	signature	sentence_1	sentence_2
e55	http://...	D290xtte5…	498efjd8…	ie939gjd…

図 9　検証サーバ内で保存するときのデータ構造
Fig.9　Data structure of DB on verification server

表 2　検証サーバ内のデータの構成要素
Table.2　Elements of data on verification server

要素名	備考
id	署名の番号
url	署名の生成元となった 2 ちゃんねるのスレッドの URL
signature	作成した署名のデータ
sentence_x	署名作成時に生成した部分文章と識別子を連結したハッシュ値

3.3.2 署名の検証

以下に検証サーバが署名の検証を行う際の手順を示す.

① ユーザのブラウザのアドオンから送信されたきた 2 ちゃんねるのスレッドの URL を用いて自身のデータベース内にすでにこの URL のスレッドの署名を過去に生成していないかどうかを調べ，生成していた場合は 3.3.1 の処理を飛ばして次の②を行う．また，生成していなかった場合には 3.3.1 の処理を行って署名を作成した後，次の②の処理を行う.

② ユーザから送信されてきたまとめサイトの URL から HTML コードを取得し，その中から「thread start」と「thread finish」のコメントアウト文で囲まれた範囲内の HTML コードを用いて 3.3.1 と同じ手順で署名を生成する．ただし，「delete writing start」と「delete writing finish」のコメントアウト文で示されている削除された部分については 3.3.1 の手順⑤で保存された *sentence_x* から対応するハッシュ値を使用することとする.

③ 3.3.1 で生成した署名あるいは検証サーバ内に保存されていた署名と②で生成した署名を比べて，検証に成功した場合には valid とまとめサイトの HTML コード内にあった書き込みの番号を，失敗した場合には invalid をユーザのブラウザのアドオンに対して送信する.

4. まとめ

本稿では，Web 上の掲示板サイトとそのまとめサイト間での署名検証および，まとめサイト内での検索性の工場を目的としたシステムについて提案し，概要を述べた．今回提案した手法については今後も議論を進めていく必要があると考えている.

参考文献

1) 2NN 2 ちゃんねるニュース速報+ナビ (http://www.2nn.jp/)
2) ニュース 2 ちゃんねる(http://news020.blog13.fc2.com/)
3) 増渕孝延，中村 創，石井真之：内部不正者を考慮した墨塗り箇所変更可能方式の提案，電子情報通信学会技術研究報告，SITE，技術と社会・倫理 105(192)，179-186，2005.
4) 武仲正彦，吉田考司，金谷延幸：検証者が署名者と墨塗り者を検証可能な電子文書の墨塗り方式，CSS 2004，6C-3，pp.475-480，2004.
5) 渡辺知恵美，新井裕子：データベースアウトソーシングにおけるプライバシ保護に考慮した範囲検索法，電子情報通信学会 第 19 回データ工学ワークショップ，C1-1，2008.