ライトワンス文書管理システム

原田篤史[†1] 西垣正勝[†2] 曽我正和[†3]
田窪昭夫[†4] 中村逸一[†5]

様々な場面で使用されるようになってきた電子文書には，追記によってのみ更新が許されるライトワンス型の文書が含まれており，電子カルテなどはその好例である．ライトワンス文書では，たとえ文書中に誤りが発見されたとしてもすでに書かれている記述を削除することが許されず，修正内容を追記として付加するしかない．ライトワンス文書を電子文書として扱う際には，データの完全性と追記の順序性の保証が必要となる．本論文は，特別なハードウェアに頼ることなく，電子化されたライトワンス文書の安全な管理を実現する方式を提案する．本方式では，追記データ単位で当該データを作成したユーザのディジタル署名と文書を保存する文書管理システムのディジタル署名を２重に付加することにより，データの完全性を保証している．同時に，文書中の各追記データにおけるユーザとシステムのディジタル署名はそれらすべてをリンキングさせることにより，追記データの順序性を保証するとともに，文書改竄に対する耐性を強化している．タイムスタンプサーバなどに代表される従来のリンキング方式とは異なり，文書ごとに独立した「文書本位」の署名のリンキングをすべてのユーザと文書管理システムが積極的にリンキングに協力する方式によって行うことで，文書の改竄困難性が非常に高く，特にライトワンス文書の管理に優れたシステムを実現している．本方式は既存の標準的な署名アルゴリズムにより実装が可能で，かつ，異なった署名アルゴリズムを用いる複数のユーザが１つの文書に対して追記することもでき，運用上の柔軟性をあわせ持つ．

A Write-once Data Management System

Atsushi Harada,[†1] Masakatsu Nishigaki,[†2] Masakazu Soga,[†3]
Akio Takubo[†4] and Itsukazu Nakamura[†5]

There are documents which require to be stored in a write-once format, e.g., medical records. Such documents can be updated only by adding new records. Even when any erratum is found, a document is not allowed to be written over, instead, new record for errata is added to the document. Moreover, it is also required to keep the order of records in a document. This paper proposes a secure database management system for a “write-once document”. In this system, two types of digital signatures are added to each record to maintain the integrity of a document. All those signatures are created with a linking scheme; the linkage between records and signatures makes it difficult for a cracker to alter/remove/insert any record or change the order of records. The system proposed here employs a “document-oriented” linking scheme with both of users’ and system’s digital signatures, whereby the system realizes a secure write-once data management system with no special hardware. In addition, this system does not depend on a particular signature algorithms. That is, a variety of users each of who uses different signature algorithms can add new records to any document in the system.

†1 静岡大学大学院情報学研究科
Graduate School of Information, Shizuoka University
†2 静岡大学情報学部
Faculty of Information, Shizuoka University
†3 岩手県立大学ソフトウェア情報学部
Faculty of Software and Information Science, Iwate Prefectural University
†4 東京電機大学情報環境学部
Faculty of Information Environment, Tokyo Denki University
†5 株式会社 NTT データセキュリティビジネスユニット
Security Business Division, NTT Data Corp.

1. はじめに

近年，様々な場面で電子化された文書が利用されるようになってきている．そのような文書には，電子カルテや稟議書のように，追記によってのみ更新が許されるライトワンス型の文書が存在する．ライトワンス文書においては，たとえ文書中に誤りが発見されたとしてもすでに書かれている記述を削除することが許されず，修正内容を追記として付加するしかない．このような性質上，ライトワンス文書は何かの証拠物とし

て保管されることも多く，その管理には高いセキュリティが要求される．

ライトワンス文書は，複数の追記データが集まって1文書が構成される．1文書に対して複数のユーザがライト権限を持つことも多い．電子的に保存される場合，各追記データは，データ改竄，削除，順序の入れ替え，偽データの挿入などから保護されなければならない．すなわち，データの完全性と追記の順序性の保証が必要となる．

上記の要求を満たすには，適切なユーザ認証およびアクセスコントロールを実施することが不可欠である．しかし，ユーザ認証のみの対処では，成りすましの問題やセキュリティホールをついてシステムに侵入される可能性が存在するため，公的な信頼性が得られない．さらに，認証を通過できる正規ユーザによる不正を許してしまう可能性もある．そこで，ユーザ認証に加えて，各データに作成者のディジタル署名を付加することが必要であろう．ディジタル署名を用いることにより，データ改竄を検出することが可能となり，公的な信頼性を得ることもできる．

ただし，ディジタル署名によって防止できるのは秘密鍵を持たない不正者によるデータの改竄であることに注意しなければならない．すなわち，秘密鍵を有する正規ユーザであれば，自らが作成した署名付きデータを改竄し，ディジタル署名を付加し直すことが可能である．よって，たとえば電子カルテの場合であれば，医師が過去の診断記録を改竄したうえでディジタル署名を付加し直し，誤診の痕跡を抹消するようなことが可能である．

このため，既存の電子カルテシステムなどではCD-Rなどのライトワンス媒体に定期的に文書を保存する方式が採用されている[4]．しかし，やはりハードウェア的な管理は煩雑であり，また，ライトワンス媒体から読み出された後のデータの信頼性を保証するためにも，ソフトウェア的にデータベース中のライトワンス文書の正当性を完全に保証する何らかの仕組みが必要であると考える．

そこで本論文では，追記データごとに当該データ作成者と文書管理システムが逐次，2重にディジタル署名を付加することによりライトワンス文書の正当性を保証する方式を提案する．追記データ作成者の署名とシステムの署名は次々とリンクされるように署名がなされるため，ある追記データを作成した本人が自らのデータを改竄して改竄後のデータにディジタル署名を付加し直したとしても，その直後に付加されているシステムの署名やそれ以降に他のユーザによって追記されたデータの署名までをも偽造することは難しく，完全な改竄（後から誰かがディジタル署名を検査しても改竄が検出されない）は不可能に近い．

ライトワンス文書の保護をデータの原本性保護ととらえると，公証サーバやタイムスタンプサーバを用いたシステム[1)~3),5)~7)]によるライトワンス文書の管理も可能である．特に，リンキングプロトコルに基づくタイムスタンプシステムは本方式との類似性が高い．また，ディジタル署名のリンキングという観点からは，本方式はヒステリシス署名[8)]と同じコンセプトの署名方式と考えることも可能である．本方式とこれらの方式の詳細な比較は6章に示すが，概念的には，データのリンキングに対するポリシーが大きく異なっていると考える．リンキングプロトコル型タイムスタンプシステムは，不特定多数のユーザがタイムスタンプサーバにデータを送り，タイムスタンプサーバは送られてきた様々な文書に順次リンクさせながらハッシュあるいは署名を施す．すなわち「サーバ本位のリンキング」を行っているといえる．一方，ヒステリシス署名は自らの署名に自らの過去の署名履歴を利用するものであり，すなわち「ユーザ本位のリンキング」といえよう．これらに対して，本方式は文書単位で不特定多数のユーザによる追記データがリンクされる「文書本位のリンキング」を実現するものである．換言すれば，本方式はライトワンス文書データベースの管理に特化した署名方式であり，文書の完全性を保証するのに適した方式であると考えられる．

文書本位のリンキングを実装する署名アルゴリズムには，他に文献9)の多重署名があげられる．多重署名は1つの文書に複数のユーザが署名を行うことが可能であり，ライトワンス文書に対する署名方式に要求されるMessage flexibility，Order flexibility，Message verifiability，Order verifiabilityのすべての性質を兼ね備えた完全性の高い署名アルゴリズムである．よって，ライトワンス文書管理システムを構築するにあたり多重署名アルゴリズムを採用することは，非常に理にかなった選択であるといえよう．

ただし，多重署名は暗号学的に署名アルゴリズムそのものを高機能化させることにより実現された署名方式である．一方，たとえばヒステリシス署名では，過去の署名文をも今回の署名に利用するという「運用上の一工夫」が従来の署名にアリバイ（暗号ブレイク後に及ぶ耐性）能力を与えている．よって著者らは，暗号要素技術そのものを強化するアプローチに限定するのではなく，暗号要素技術の適切な運用方法を検討することも重要であると考える．本論文で提案している

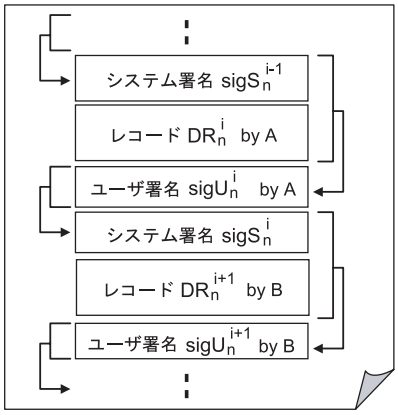

図 1 文書 $DO_n^{t_n}$ の構成
Fig. 1 Format of a document.

2 重の署名による追記型文書の管理方式は，既存の標準的な署名アルゴリズムを（アルゴリズムには手を加えることなく）効果的に運用することにより，ライトワンス文書管理システムを構築するための一方法である．

2. ライトワンス文書管理システム

2.1 文書データ

本方式では，システムが保管する n 番目の「文書」を $DO_n^{t_n}$ で表す．ここで，t_n は n 番目の文書の「カウンタ情報」である．$t_n = i$ において，あるユーザが文書 DO_n^{i-1}（文書が上書きされた時点でカウンタ情報がインクリメントされるため，$t_n = i$ でシステムに保存されている文書は DO_n^{i-1} である）を読み，これに追記したとする．ここで，ユーザが追記した実際の記述内容を「レコード」と呼び，$DR_n^{t_n}$ で表す．DO_n^{i-1} に追記されるレコードは DR_n^i である．ユーザは DO_n^{i-1} に含まれる最新のシステムの署名と DR_n^i とを連結したデータに署名を付し，システムに保存を依頼する．ここで，ユーザによって付加される署名を「ユーザ署名」と呼ぶこととし，$sigU_n^{t_n}$ と表す．システムは DR_n^i に付加されたユーザ署名 $sigU_n^i$ を確認し，それが正当であると認められた場合に限り DR_n^i を認可し，ユーザ署名 $sigU_n^i$ に対してシステムの署名を施す．ここで，システムによって付加される署名を「システム署名」と呼び，$sigS_n^{t_n}$ と表す．そして，今までの DO_n^{i-1} に DR_n^i，$sigU_n^i$，$sigS_n^i$ を付加したデータの集合を，新たな DO_n^i としてシステム内のデータベースに保存する．すなわち，文書の更新は，レコードが次々と追記されていくことによりなされる．この様子を**図 1** に示す．ただし，文書の新規作成時（$t_n = 1$ の時点）には，追記対象の文書が存在しないため，ユーザ署名 $sigU_n^1$ はユーザが記述したレコード DR_n^1 のみから生成される．

システムおよび各ユーザは，公開鍵暗号の秘密鍵と公開鍵のペアを持つ．以下，システムの秘密鍵を Q，公開鍵を P とし，ユーザ x の秘密鍵を q_x，公開鍵を p_x とする．システムおよび各ユーザは，自らの秘密鍵を用いて任意のデータにディジタル署名を施すことができ，署名者の公開鍵を用いることで任意の署名を検証することができる．以下，データ D を秘密鍵 Q，q_x で署名したものをそれぞれ $Q(D)$，$q_x(D)$ と記す．

文書データの構成をまとめると，以下の式のようになる．ここで，$\|$ はデータの連結を意味する．

$$
\begin{aligned}
&t_n = 1\\
&\quad DO_n^1 = \{DR_n^1 \| sigU_n^1 \| sigS_n^1\}，\text{ただし}\\
&\qquad sigU_n^1 = q_x(DR_n^1)\\
&\qquad sigS_n^1 = Q(sigU_n^1)\\
&t_n \geq 2\\
&\quad DO_n^i = \{DO_n^{i-1} \| DR_n^i \| sigU_n^i \| sigS_n^i\}，\text{ただし}\\
&\qquad sigU_n^i = q_x(sigS_n^{i-1} \| DR_n^i)\\
&\qquad sigS_n^i = Q(sigU_n^i)
\end{aligned}
$$

本方式において，文書は複数のレコードおよび署名がひとまとまりになったものである．各レコードには必ずユーザ署名およびシステム署名が 2 重に付加されている．ユーザ署名は追記レコードと追記対象の文書における最後尾のシステム署名から計算される．一方，システム署名はユーザ署名から算出される．この結果，文書内のすべてのユーザ署名およびシステム署名は完全にリンク付けされる．

文書のカウンタ情報 t_n は，文書の更新（追記）がシステムによって認可されるごとに 1 つ進められる．カウンタ情報を参照することで，対応する文書 $DO_n^{t_n}$ のバージョンを知ることができる．

2.2 構　　成

本方式の構成は**図 2** のようになる．システムは各文書をそれぞれのカウンタ情報とともに保管する．ユーザ A〜Z はシステムの正規ユーザであり，ネットワークを介してシステムに接続している．各ユーザはシステムが保管する文書に対して閲覧および追記が可能であり，かつ，新たな文書を新規作成することができる．システムは自身にアカウントを持つすべてのユーザの公開鍵を管理する．

ユーザがシステムにアクセスする際には，必ず，システムによってユーザ認証が行われる．すなわち，システムにアカウントを持たないユーザはどの文書にも

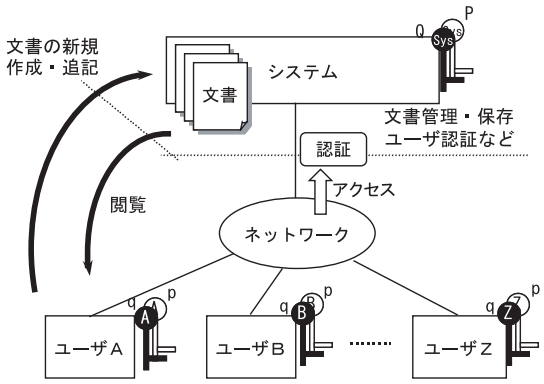

図 2 システムの構成
Fig. 2 System configuration.

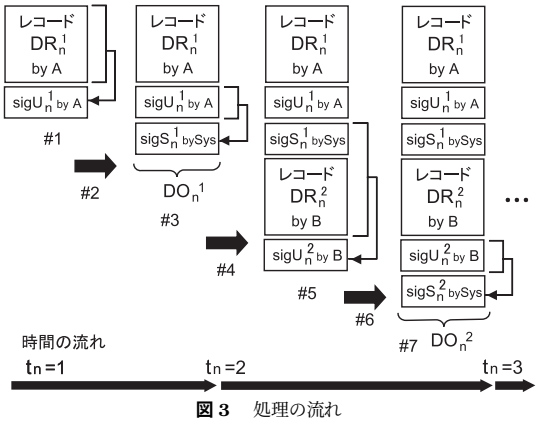

図 3 処理の流れ
Fig. 3 A document at each step of the procedure.

アクセスできない．さらに，システムにアカウントを持つ正規のユーザであっても編集する権限を持たない文書にはアクセスできないように，適切なアクセスコントロールを施すような運用を行うことも可能である．ただし，不正者がクラッキングによりシステムに侵入することは当然起こりうる．

2.3 文書に対する処理の手順

本方式における文書の新規作成，閲覧，追記の処理手順を説明する（**図 3**）．以下の手順において，あるユーザがある文書に対する追記を行う際には，処理が開始されてから完了するまでの間，他のユーザが当該文書に対して追記を行うことができないように，システムによって排他制御が行われるものとする．

#1 ユーザ A が文書を新たに作成するとする．新規作成であるので，文書 $DO_n^{t_n}$ はこの時点ではシステムのデータベースに存在しない．新規文書の場合，カウンタ情報は $t_n = 1$ である．ユーザ A がレコード DR_n^1 を書き，自らの秘密鍵 q_A を用いてユーザ署名 $sigU_n^1$ を生成する．

$$t_n \leftarrow 1 \tag{1}$$
$$sigU_n^1 = q_A(DR_n^1) \tag{2}$$

#2 ユーザ A はレコード DR_n^1 とこれに対応するユーザ署名 $sigU_n^1$ とをシステムに送信する．

#3 システムは受け取った DR_n^1 と $sigU_n^1$ の整合性をユーザ A の公開鍵 p_A を用いて検証する．問題がなければ，自らの秘密鍵 Q を用いてユーザ署名 $sigU_n^1$ からシステム署名 $sigS_n^1$ を計算し，これらを結合した $\{DR_n^1 \,\|\, sigU_n^1 \,\|\, sigS_n^1\}$ を文書 DO_n^1 としてデータベースに保存する．この時点でカウンタ情報を $t_n = 2$ とする．

$$sigS_n^1 = Q(sigU_n^1) \tag{3}$$
$$DO_n^1 = \{DR_n^1 \| \, sigU_n^1 \| \, sigS_n^1\} \tag{4}$$
$$t_n \leftarrow 2 \tag{5}$$

#4 次に，別のユーザ B が文書 DO_n^1 を閲覧し，追記を加えたいとする．

#5 ユーザ B は追記分のレコード DR_n^2 を書き，自らの秘密鍵 q_B を用いて，DO_n^1 中の最後尾のシステム署名 $sigS_n^1$ と自らが記述した DR_n^2 とを連結したデータに対してユーザ署名 $sigU_n^2$ を生成する．

$$sigU_n^2 = q_B(sigS_n^1 \| \, DR_n^2) \tag{6}$$

#6 ユーザ B は DR_n^2 と $sigU_n^2$ とを，文書 DO_n^1 に対する追記としてシステムに送信する．

#7 システムは受け取った $\{DR_n^2 \| \, sigU_n^2\}$ と今までの文書 DO_n^1 の正当性（DR_n^2 の整合性および連続性）を検証する．問題がなければ，システムは当該追記レコードにおけるユーザ署名 $sigU_n^2$ に対してシステム署名 $sigS_n^2$ を生成する．そして，これら全体を結合した $\{DO_n^1 \| \, DR_n^2 \| \, sigU_n^2 \| \, sigS_n^2\}$ を文書 DO_n^2 としてデータベースに保存する．この時点でカウンタ情報を $t_n = 3$ とする．

$$sigS_n^2 = Q(sigU_n^2) \tag{7}$$
$$DO_n^2 = \{DO_n^1 \| \, DR_n^2 \| \, sigU_n^2 \| \, sigS_n^2\} \tag{8}$$
$$t_n \leftarrow 3 \tag{9}$$

#8 他のユーザ x（ユーザ A，ユーザ B を含む）がさらにこの文書に追記したい場合には，#4 から同様の手順を繰り返す．

$t_n = i (i = 3, 4, 5, \cdots)$ として，

$$sigU_n^i = q_x(sigS_n^{i-1} \| \, DR_n^i) \tag{10}$$
$$sigS_n^i = Q(sigU_n^i) \tag{11}$$
$$DO_n^i = \{DO_n^{i-1} \| \, DR_n^i \| \, sigU_n^i \| \, sigS_n^i\} \tag{12}$$
$$t_n \leftarrow i + 1 \tag{13}$$

文書 $DO_n^{t_n}$ 中のすべてのレコード $DR_n^{t_n}$ には必ず当該レコード作成者によるユーザ署名およびシステム署名が 2 重に付加されていること，および，すべての

署名データが完全にリンキングされており，文書全体をカバーしていることに注目されたい．

3. 考　　察

3.1 攻撃耐性

文書 $DO_n^{t_n}$ 中のすべてのレコード $DR_n^{t_n}$ にはレコード作成者のディジタル署名（ユーザ署名）が付加されており，他の任意のユーザによる当該レコードの改竄を防止する．加えて，レコード作成者のディジタル署名には必ず文書管理システムによる署名（システム署名）が付加されており，レコード作成者本人によるレコードの改竄も防止する．

文書 $DO_n^{t_n}$ 中のすべてのユーザおよびシステム署名は，完全にリンクされている．このため，たとえレコード $DR_n^{t_n}$ を作成したユーザがシステムの秘密鍵を知ることができたとしても，レコードを完全に改竄することはできない．なぜなら，あるレコードに手を加えると，そのレコード以降に付加されたすべての署名を偽造する必要が生じ，他の多数のユーザの秘密鍵を盗むことが要求されるからである．同様に，レコード $DR_n^{t_n}$ の順序の入れ替え，削除，偽データの挿入なども防止される．

攻撃者が，システムの秘密鍵を知ることができ，かつ，改竄部分以降のレコードを追記したすべてのユーザの秘密鍵を取得できた場合のみ，レコードの改竄（後から誰かがディジタル署名を検査しても改竄が検出されない）が可能となる．しかしながら，多数の秘密鍵を盗みだすことには相当な労力がともなうと考えられ，文書の改竄は非常に困難であると思われる．本システムの文書は，追記の数が多ければ多いほど改竄に対する耐性が向上する．

悪質な攻撃者は，システム内の文書の改竄はできないとしても，システムに侵入して文書全体を破壊したり，文書を古い文書に書き戻したりする（文書 DO_n^i に対して DO_n^{i-1}, DO_n^{i-2}, $\cdots$ が古い文書である）攻撃を行うかもしれない．よって，本方式においては，システム自身の攻撃耐性を強化することに加え，本方式をバックアップシステムと連携して運用することも重要であると考えられる．そこで，4.1 節において本方式の攻撃耐性をさらに向上させる方法について述べるとともに，4.2 節ではバックアップ方式の一例について説明する．

3.2 効　率　性

新しいレコードが追記されるとき，追記レコードに対するユーザ署名は，追記対象の文書の最後尾に存在しているシステム署名のみを当該追記レコードに連結したデータより算出される．そして，これに続いて生成されるシステム署名は，当該ユーザ署名のみから計算される．したがって，追記が行われるにつれ文書全体が大きくなっていっても，1 回の署名生成にかかるコストはほとんど変化しない．

一方，文書の完全性を検証する際には，追記の数が増えて文書が大きくなっていくほど検証コストが高くなっていく．これは，文書の検証を行うときは，その文書に含まれる最新のシステム署名から，新しい順にすべてのレコードに対するシステム署名およびユーザ署名をリンクを遡って検証しなくてはならないからである．しかし，文書の検証は，ユーザが当該文書を読んでいる間にバックグラウンドプロセスとして実行すればよい．または，業務が行われない深夜などの時間帯にシステム内の文書の完全性チェックをまとめて行い，通常業務時は簡易的なチェック（リンクをたどらずに必要なレコードのユーザ署名とシステム署名のみをチェックする）で済ませる，という運用による対処も可能である．

3.3 運用の柔軟性

本論文は，通常の署名アルゴリズムを運用により高機能化することを目的としている．すなわち，2 章に示した 2 重署名のリンキングを用いることにより，任意の標準的な署名アルゴリズムによってライトワンス文書管理システムの構築が可能であることを示すものである．

通常の署名を運用により高機能化するというアプローチは，異なる署名アルゴリズムのシステムの連携運用を可能にするという意味でも有効である．すなわち，本ライトワンス文書管理システムにおいては，異なる署名生成アルゴリズムを持つ複数のユーザが 1 つの文書に順次追記をしていくことも可能である．**図 4** に，1 つの文書に 1,024 ビット RSA 署名のユーザと 1,024 ビット ElGamal 署名のユーザが追記を行った（システムは 2,048 ビット RSA 署名）例を示す．ユーザ署名，システム署名の各々に付けられたヘッダ情報から，検証者は当該署名データの署名アルゴリズムを知ることができるようになっている．

このため，様々な CA が独自のポリシー（採用している公開鍵暗号も異なる）で運営されており，ユーザがすでにいずれかの CA から秘密鍵と公開鍵証明書を発行してもらっている場合，ユーザはシステムにその公開鍵証明書を登録しさえすれば，自分が今まで使用していた秘密鍵と署名アルゴリズムによりライトワンス文書を作成する（追記文書に署名を施す）ことができる．ただし，文書の検証のために，システムは各

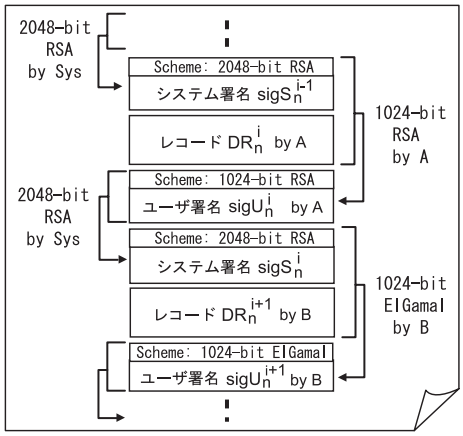

図 4 異なる署名方式によって追記された文書
Fig. 4 A document signed with various signature schemes.

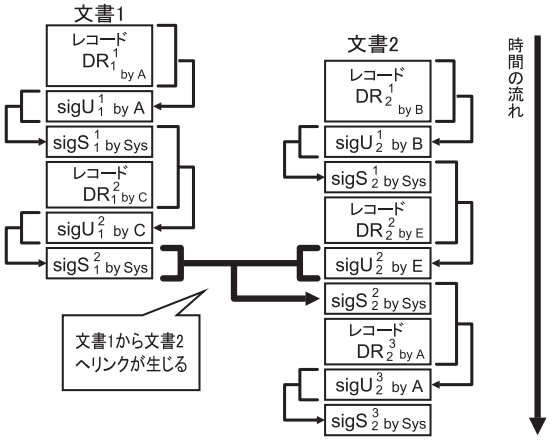

図 5 署名交差
Fig. 5 Signature intercrossing.

ユーザの公開鍵に加え，すべてのユーザに対する署名検証アルゴリズムを備える必要がある．

ユーザがある時点で CA を変更（すなわち，署名アルゴリズムを変更）したり，CA が安全性の拡張などの理由で暗号方式を変更したりした場合にも，本方式では当該ユーザの署名アルゴリズムのみを更新するだけでよく，柔軟性のある運用が可能となる．

4. 付加的機能

4.1 署名交差

3.1 節で述べたように，本方式においては文書に多くの追記が加えられているほどその文書の改竄に対する耐性が向上する．しかし，すべての文書に対して追記が頻繁に行われるとは限らない．追記が少ない文書や追記されたばかりのレコードはそれ以降に追記されているレコードが比較的少ないため，クラッカーは少数の秘密鍵の解読により改竄が可能になる．ここで，本方式がヒステリシス署名と類似性を有することから，追記の少ない文書の改竄困難性を高めるためにヒステリシス署名の履歴交差[8]を利用可能であることが予想される．本方式においては，同様の操作を「署名交差」と呼んでいる．以下にその具体的な処理および効果を簡単に示す．

図 5 は，追記が頻繁に行われる文書 2 に対する署名生成の際に，その署名対象データに追記が少ない文書 1 中の署名を含めることによって，文書 1 から文書 2 へと新たにリンクが追加される例を示している．

図 5 の例において，文書 1 の 1 番目のレコード DR_1^1 の部分を完全に（後から誰かがディジタル署名を検査しても改竄が検出されいように）改竄したい場合，文書 1 中の DR_1^1 以降に付加された全レコードに対するユーザ署名・システム署名をすべて偽造することはもちろん，さらに，文書 2 における 2 番目のレコード DR_2^2 に対するシステム署名およびそれ以降のレコードに対するユーザ署名・システム署名もすべて偽造しなくてはならない．このように，追記レコードの数が少ない文書に対しても，署名交差によって文書の改竄を困難にすることが可能である．

なお，図 5 は 2 文書間の署名交差を示した例であるが，同様の仕組みで多数の文書間で幾重にも署名の交差を行うような運用も可能である．また，複数のシステム間の文書の署名を交差させる方法も考えられる．新聞紙面などに定期的に文書の署名データを公開[6),7)]するなどの方法をとっても同様の効果が期待できる．

4.2 バックアップ

本方式においては，文書を完全に（後から誰かがディジタル署名を検査しても改竄が検出されないように）改竄するには，システムの秘密鍵のほかに改竄部分以降のレコードを追記した全ユーザの秘密鍵が必要になる．したがって，システムがクラッカーの侵入を許すか，あるいはシステムの管理者が悪意を持っていたとしても，文書を完全に改竄することはできない．しかし，悪意を持つ者の目的が文書の改竄ではなく，文書の消去・破壊であった場合には，不正者はシステムに侵入することさえできればシステム内の文書データを消去したり，最新の文書を古い文書に書き戻したりすることが可能となる．

文書の破壊も改竄の一種であるため，署名交差に関係している文書が 1 つでも残っている限りディジタル署名の検査によりこのような事態は検出可能である．その際，破壊された文書を復旧するためには，本シス

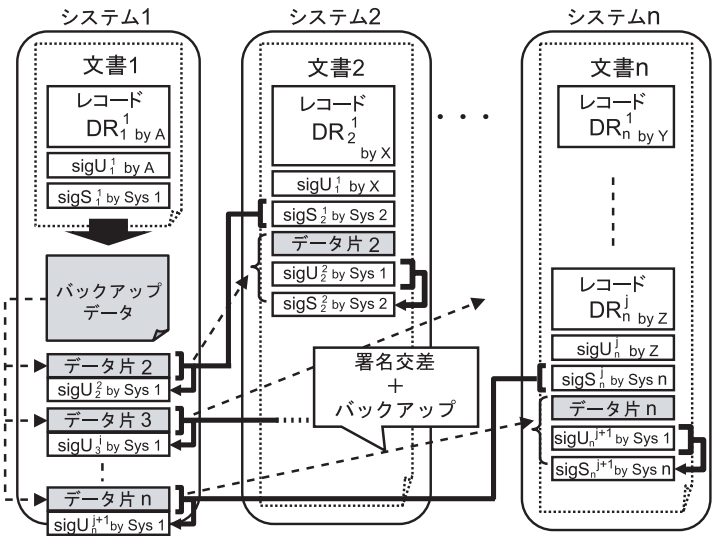

図 6 署名交差を利用した文書のバックアップ
Fig. 6 Data back-up based on signature intercrossing.

テムは文書のバックアップ機能を具備する必要がある．バックアップの方式は任意であるが，ここではその一例として，署名交差との親和性の高いバックアップ方式（**図 6**）を示す．

図 6 において，システム 1 は自身の管理する文書 1 のバックアップデータを n−1 個のデータ片 2〜n に分割する．そして，それぞれをシステム 2〜n が持つ文書 2〜n への追記レコードとして処理してもらう（データ片 k が追記レコードに相当し，データ片 k に対するシステム 1 の署名がユーザ署名に，これに続いて付されるシステム k による署名がシステム署名に対応している）．なお，文書 2〜n の表示時には，当該データ片は無視されるものとする．そして，文書 1 の破損が認められた場合には，システム 2〜n はそれぞれに分配されたデータ片をシステム 1 に提出する．システム 1 は送られてきたデータ片を結合してバックアップデータを復元し，文書 1 を復旧することができる．

本バックアップ方式を用いる場合，バックアップデータの一部が他システムおよび他ユーザの手に渡ることになるため，各データ片からの情報漏洩に対して配慮が必要である．また，文書の復元時にすべてのシステムから当該文書のバックアップデータ片を回収できないこともありうるだろう．これらの問題に対しては，バックアップデータを暗号化したり，秘密分散[10]の技法を適切に用いたりすることによって，各データ片から文書の情報が漏洩することを防ぐことや，すべてのデータ片が集まらなくてもバックアップデータを復元できるようにするなどの対策が有効である．

4.3 署名交差の柔軟性

3.3 節で述べたように，本ライトワンス文書管理システムでは，各ユーザが異なる署名アルゴリズムを用いていても問題ない．そして，これは 4.1 節で述べた署名交差に対しても同様である．すなわち，2 章に示した 2 重署名のリンキングによりライトワンス文書管理システムを実装していさえすれば，ユーザ間，システム間で異なる署名アルゴリズムを採用していても自システム内の任意の他文書や他システム内の任意の文書との間で署名交差やバックアップを行うことが可能である．

5. 第 3 者の署名によるデータの封印 ——電子カルテシステムへの応用

前章までに説明したように，本ライトワンス文書管理システムは各追記レコードごとに当該レコード作成者による署名（ユーザ署名）とシステムによる署名（システム署名）の 2 重のディジタル署名を施す．ここで重要なことは，レコード作成者以外のエンティティであるシステムのディジタル署名によって各追記レコードを封印することである．よって，レコード作成者以外の第 3 者の署名であれば，これによりシステム署名を代用することが可能である．本章では電子カルテ管理システムを例に採り，追記レコード作成者（医師）と第 3 者（患者）がカルテに対して 2 重署名を行うシステムを説明する．

電子カルテシステムの構成を，**図 7** に示す．ユーザが医師に，文書がカルテになっただけで，2 章の図 2

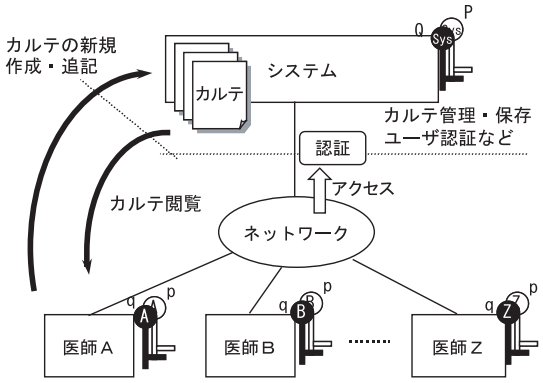

図 7 電子カルテシステムの構成
Fig. 7 System configuration for managing electronic medical records.

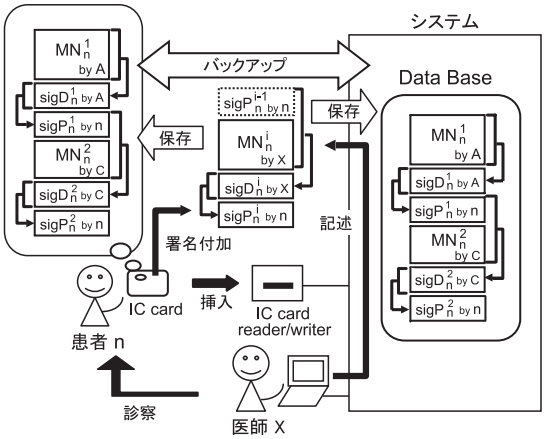

図 8 患者の署名を用いた電子カルテシステム
Fig. 8 A medical records system with patients' signatures.

で示したライトワンス文書管理システムの構成と同じ形で実現が可能である．診察のたびに記されるカルテ（レコード）に対して行われる処理も，2.3 節の手順に準ずる．したがって，本電子カルテシステムは前章までに示したライトワンス文書管理システムと同等の攻撃耐性および効率性を有する．

本電子カルテシステムの特徴は，システム署名を用いる代わりに，患者自身の署名で医師の記述したカルテの内容を封印することである．患者の署名を 2 重署名に利用した電子カルテシステムを**図 8** に示す．ここでは，各患者は所有する IC カードの中に自身の秘密鍵を保持しており，診療時に IC カードを電子カルテシステムのスロットに差込むことでカルテに患者自身のディジタル署名を付加することができるとする．

医師は患者を診察し，毎回の診察結果 $MN_n^{t_n}$（ライトワンス文書管理システムにおけるレコード $DR_n^{t_n}$ に対応）を記述したうえで医師自身のディジタル署名 $sigD_n^{t_n}$（ライトワンス文書管理システムにおけるユーザ署名 $sigU_n^{t_n}$ に対応）を付加する．さらに，そのデータに診察を受けた患者のディジタル署名 $sigP_n^{t_n}$（ライトワンス文書管理システムにおけるシステム署名 $sigS_n^{t_n}$ に対応）が付加される．$t_n = i$ における医師によるディジタル署名 $sigD_n^i$ は，直前の診察結果に対する患者の署名 $sigP_n^{i-1}$ と今回の診察において医師が新たに記述した診察結果 MN_n^i とから生成される．一方，患者の署名 $sigP_n^i$ は今回の診察結果に対する医師のディジタル署名 $sigD_n^i$ から生成される．この結果，各患者のカルテ内に記述されているすべての診察結果 $MN_n^{t_n}$ およびこれらに対するすべての署名がリンクされる．なお，患者は自身のカルテを IC カードに保持して携帯することも可能である．

なお，実際に電子カルテを運用する場合，通常，ある患者に対する主治医は固定であることが多いと考えられる．しかし，診療に際してレントゲンをとれば放射線医が，麻酔を用いれば麻酔医がその旨をカルテに記述し，署名を付加することが必要になる．こうした医師以外にも，入院患者に対しては，看護婦によって毎日の検温や投薬の経過などの看護記録が記述される．さらに，薬が出されるならば薬剤部が，レセプトの処理のためには事務局が介在し，現実には病院全体の処理とも連携した統合的なカルテシステムが構築されることになる．したがって，現実の電子カルテシステムにおいて，1 人の患者のカルテに関わる医師および医療スタッフは多数にのぼることが通常であり，本ライトワンス文書管理システムの適用が効果的である．

システム署名の代わりに患者の署名を利用する電子カルテシステムには，次のような利点がある．1) システム署名を用いていないので，たとえ不正者がシステムの秘密鍵を知ったとしても，カルテの偽造に利用することはできない．2) カルテをシステムおよび患者自身の IC カードの両方に保存するようにすれば，不正者がシステムに侵入できたとしてもカルテのデータを消去することはできず（適宜，IC カードからシステムにデータを書き戻せばよい），患者が IC カードを紛失・破損した場合にもカルテの再発行が可能である（適宜，システムから IC カードにデータを書き戻せばよい）．3) 通常，患者は医師が過去の誤診を隠すために自分のカルテを書き換えることを認めないため，患者と医師が結託することは基本的に考えられない．

6. 比較評価

我々が提案する方式を従来の類似技術と比較検討す

ることにより，本方式を評価する．比較対象は以下のとおりである．

(1) タイムスタンプ方式：リンキングプロトコルに基づくタイムスタンプサーバが運用されている．レコード作成者はレコードを記述し，これに自身の秘密鍵で（通常の）署名を付したうえで，サーバに署名付きレコードを送信する．サーバは署名付きレコードを受信するたびに，当該レコードのハッシュをリンキング方式で生成し，ユーザにこれを返信する．ユーザは，署名付き追記レコードとこれに対するサーバのハッシュを文書管理システムに送信する．

(2) ヒステリシス署名方式A：各ユーザ（レコード作成者）がヒステリシス署名生成機能を所有している．レコード作成者はレコードを記述し，これに自身のヒステリシス署名を付したうえで，文書管理システムにこれを送信する．文書管理システムは各ユーザから送られてきたヒステリシス署名付きレコードのハッシュをリンキング方式で生成し，ヒステリシス署名付きレコードとともに最新のハッシュ値を保存する．

(3) ヒステリシス署名方式B：文書管理システムがヒステリシス署名生成機能を所有している．レコード作成者はレコードを記述し，これに自身の秘密鍵で（通常の）署名を付したうえで，文書管理システムにこれを送信する．文書管理システムは署名付きレコードを受信するたびに，当該レコードのヒステリシス署名を計算し，これを署名付きレコードとともに保存する．

(4) 多重署名方式：各ユーザ（レコード作成者）および文書管理システムが文献9)の多重署名生成機能を所有している．レコード作成者は，レコードを記述し，文献9)のアルゴリズムで署名を行うことにより文書にレコードをリンキングする．ユーザがレコードを追記するたびに，システムが文献9)のアルゴリズムにより受領確認のためのレコードを追記し，ユーザの追記を封印する．

(5) 本方式：レコード作成者はレコードを記述し，現在の文書の最後尾のレコードのシステム署名と自身が記述したレコードからユーザ署名を計算する．文書管理システムは，ユーザからユーザ署名付きレコードを受信するたびに，当該ユーザ署名に対するシステム署名を計算し，これをレコードおよびユーザ署名とともに保存する．

上記ヒステリシス署名方式AおよびBについて，文書間における履歴交差は行わないものとする．これは，ヒステリシス署名の履歴交差の具体的な運用法（どれくらいの頻度でどのエンティティと履歴交差を行うのが必要十分であるか）は文献8)にも明確には記されておらず，今後の研究が待たれる部分だからである．これは我々の署名交差においても同様であり，よってここでは，本方式においても署名交差を用いないシステムを対象として比較評価を行うこととした．ただし，6.4節において，運用の柔軟性を比較するにあたり，履歴交差・署名交差の可能性については言及する．なお，比較の結果は**表1**にまとめた．

6.1 リンキングと検証

(1) タイムスタンプ方式：タイムスタンプサーバは，不特定多数のユーザからの任意の文書のレコードを受け取る．すなわち，レコードのリンキングは「サーバ本位」でなされている．このため，ある1文書中の一連のレコードの正当性を検証する際に，サーバがリンクした順に複数の文書のレコードを遡って検査しなければならない．

(2) ヒステリシス署名方式A：各レコードは，レコード作成者個々人のヒステリシス署名によってリンキングされている．すなわち,「ユーザ本位」でのリンキングが行われている．さらに，レコードの順序を保証するために，文書管理システムによって各レコードのハッシュがリンキング方式で生成され，最新のハッシュ値が保存されている．したがって，ハッシュのリンキングに対しては「システム本位」のリンキングがなされている．このため，ある1文書中の一連のレコードの正当性を検証する際に，文書中のすべてのレコードの作成者のヒステリシス署名をすべて検査しなければならず，さらに，レコードの順序の正当性をハッシュのリンキングを遡って検証しなければならない．

(3) ヒステリシス署名方式B：各レコードのリンキングは文書管理システムのヒステリシス署名により実現されている．システムは，不特定多数のユーザから任意の文書のレコードを受け取る．すなわち，レコードのリンキングは「システム本位」でなされている．この場合，ある1文書中の一連のレコードの正当性を検証するために，システムのヒステリシス署名をすべて検査しなければならない．もちろん，システムは文書ごとに独立してヒステリシス署名を適用するというような運用も可能であり，この場合は「文書本位」のリンキングとなる.（今回は履歴交差は

表 1 従来方式との比較
Table 1 A comparison with conventional schemes.

	方式の概要(リンキングの特徴)	1文書の完全性検証に要するコスト	レコードの改竄を行うにあたってのハードル	レコードの入替を行うにあたってのハードル	外部通信	柔軟性
タイムスタンプ方式	・ユーザはレコードに通常の署名を付したうえでタイムスタンプサーバに送信 ・サーバは受信データのハッシュをリンキング方式で生成して返信(サーバ本位のリンキング) ・文書管理システムは全ての署名付きレコードと対応するハッシュを保存	効率: △ ・文書中の全レコードに対する署名の検証 ・サーバがリンクした不特定多数の文書レコードを遡ってのハッシュの検証 (対象文書と関係ないレコードに対するハッシュの検証が必要)	強度: ○ ・レコード作成者の秘密鍵取得 ・当該レコード改竄と署名の再計算 ・タイムスタンプサーバへの侵入 ・ハッシュのリンクの再計算 ・文書管理システムへの侵入 ・当該レコードの書き換え	強度: △ ・タイムスタンプサーバへの侵入 ・ハッシュのリンクの再計算 ・文書管理システムへの侵入 ・当該レコードの書き換え	必要	高
ヒステリシス署名方式A	・ユーザはレコードにヒステリシス署名を付したうえで文書管理システムに送信(ユーザ本位のリンキング) ・システムはヒステリシス署名付きレコードを受信するたびに当該レコードのハッシュをリンキング方式で生成(システム本位のリンキング) ・システムはすべてのレコードと対応するヒステリシス署名,および最新のハッシュを保存	効率: × ・文書中の全レコードの作成者ごとに,当該ユーザの過去のすべてのヒステリシス署名の検査 ・ハッシュのリンクの検証 (対象文書と関係ないレコードに対するヒステリシス署名およびハッシュのリンクの検証が必要)	強度: ○ ・レコード作成者の秘密鍵取得 ・レコード作成者の署名履歴取得 ・当該レコード改竄とヒステリシス署名の再計算 ・文書管理システムへの侵入 ・当該レコードの書き換えとハッシュのリンクの再計算	強度: × ・文書管理システムへの侵入 ・当該レコードの書き換えとハッシュのリンクの再計算	不要	高
ヒステリシス署名方式B	・ユーザはレコードに通常の署名を付したうえで文書管理システムに送信 ・システムは署名付きレコードを受信するたびに,当該レコードに対するヒステリシス署名を生成(システム本位のリンキング) ・システムはすべてのレコードと対応する署名,およびヒステリシス署名を保存	効率: △ ・文書中の全レコードに対する署名の検証 ・システムが過去に計算したすべてのヒステリシス署名の検査 (対象文書と関係ないレコードに対するヒステリシス署名の検証が必要)	強度: ○ ・レコード作成者の秘密鍵取得 ・当該レコード改竄と署名の再計算 ・文書管理システムへの侵入 ・文書管理システムの秘密鍵取得 ・文書管理システムの署名履歴取得 ・当該レコードの書き換えとヒステリシス署名の再計算	強度: ○ ・文書管理システムへの侵入 ・文書管理システムの秘密鍵取得 ・文書管理システムの署名履歴取得 ・当該レコードの書き換えとヒステリシス署名の再計算	不要	高
	・ユーザはレコードに通常の署名を付したうえで文書管理システムに送信 ・システムは署名付きレコードを受信するたびに,「各文書ごとに独立に」当該レコードのヒステリシス署名を生成(文書本位のリンキング) ・システムはすべてのレコードと対応する署名,およびヒステリシス署名を保存	効率: ○ ・文書中の全レコードに対する署名の検証 ・文書中の全レコードに対するヒステリシス署名のみの検証 (ヒステリシス署名のリンクは文書ごとに閉じているため)	強度: ○ ・レコード作成者の秘密鍵取得 ・当該レコード改竄と署名の再計算 ・文書管理システムへの侵入 ・文書管理システムの秘密鍵取得 ・文書の署名履歴は当該文書の全レコードに等しいので取得可能 ・当該レコードの書き換えとヒステリシス署名の再計算	強度: △ ・文書管理システムへの侵入 ・文書管理システムの秘密鍵取得 ・文書の署名履歴は当該文書の全レコードに等しいので取得可能 ・当該レコードの書き換えとヒステリシス署名の再計算	不要	高
多重署名方式	・ユーザはレコードを記述し,文献9)の多重署名を計算 ・ユーザは多重署名付きレコードを文書管理システムに送信 ・システムはすべてのレコードに対して受領確認のためのレコードを追記し文献9)の多重署名を生成 ・システムはすべてのレコードと対応する多重署名を保存 (すべてのユーザとシステムが関わる文書本位のリンキング)	効率: ○ ・文書中の全レコードに対する署名の検証 (多重署名は文書ごとに閉じているため)	強度: ◎ ・レコード作成者の秘密鍵取得 ・当該レコード改竄と多重署名の再計算 ・文書管理システムへの侵入 ・文書管理システムの秘密鍵取得 ・当該レコード以降の全追記者の秘密鍵取得 ・当該レコード以降のすべての多重署名の再計算 ・文書の書き換え	強度: ◎ ・文書管理システムへの侵入 ・文書管理システムの秘密鍵取得 ・当該レコード以降の全追記者の秘密鍵取得 ・当該レコード以降のすべての多重署名の再計算 ・文書の書き換え	不要	低
本方式	・ユーザはレコードを記述し,現在の文書の最後尾のシステム署名と自身が記述したレコードからユーザ署名を計算 ・ユーザはユーザ署名付きレコードを文書管理システムに送信 ・システムはユーザ署名付きレコードを受信するたびに,当該ユーザ署名に対するシステム署名を計算 ・システムはすべてのレコードと対応するユーザ署名およびシステム署名を保存 (すべてのユーザとシステムが関わる文書本位のリンキング)	効率: ○ ・文書中の全レコードに対するユーザ署名とシステム署名のみの検証 (ユーザ署名とシステム署名のリンクは文書ごとに閉じているため)	強度: ◎ ・レコード作成者の秘密鍵取得 ・当該レコード改竄とユーザ署名の再計算 ・文書管理システムへの侵入 ・文書管理システムの秘密鍵取得 ・当該レコードのシステム署名の再計算 ・当該レコード以降の全追記者の秘密鍵取得 ・当該レコード以降の全ユーザ署名およびシステム署名の再計算 ・文書の書き換え	強度: ◎ ・文書管理システムへの侵入 ・文書管理システムの秘密鍵取得 ・当該レコード以降の全追記者の秘密鍵取得 ・当該レコード以降のすべてのユーザ署名およびシステム署名の再計算 ・文書の書き換え	不要	高

考慮しないため)文書ごとにリンクは閉じており,各文書中の一連のレコードの正当性の検証を効率的に行うことが可能である.また,個々の文書で独立したリンキングを行うことは,事故あるいは故意により署名データが破壊されてしまった場合でも,その影響が 1 文書のみにとどまり,他文書の検証の障害にはならないという利点もあわせ持っている.

(4) 多重署名方式:レコードのリンキングは「文書本位」でなされている.すなわち,ある文書中の署名に対するリンキングは当該文書中で閉じており,各文書中の一連のレコードの正当性検

証を効率的に行うことが可能である．また，個々の文書で独立したリンキングを行うことは，事故あるいは故意により署名データが破壊されてしまった場合でも，その影響が 1 文書のみにとどまり，他文書の検証の障害にはならないという利点もあわせ持っている．

(5) 本方式：レコードのリンキングは「文書本位」でなされている．すなわち，ある文書中の署名に対するリンキングは（今回は署名交差は考慮しないため）当該文書中で閉じており，各文書中の一連のレコードの正当性の検証を効率的に行うことが可能である．署名データが破壊されてしまった場合の利点についても多重署名方式と同様である．

6.2 攻撃耐性

(1) タイムスタンプ方式：各レコードの内容そのものは各レコード作成者のディジタル署名により守られている．各レコードのリンキングはタイムスタンプサーバが行うハッシュ化によってなされている．ハッシュ値は誰でも計算が可能である．したがって，不正者がサーバに侵入することができれば，過去のすべてのハッシュを計算し直し，サーバ内のハッシュのリンク情報を改竄することができる．ただし，各レコードの内容そのものは各レコード作成者のディジタル署名により守られているため，不正者はレコードの内容を変更することはできない．よって，その後，不正者が文書管理システムへの侵入にも成功した場合，不正者はレコードの順序の入れ替えが可能となる．不正者がさらにあるレコード作成者の秘密鍵をも取得することができた場合，当該レコード作成者によって追記されたレコードに関してはその改竄も可能となる．

(2) ヒステリシス署名方式 A：各レコードは，レコード作成者個々人のヒステリシス署名によって守られている．しかし，レコードの順序はシステムによるハッシュのリンキングによって保護されているのみである．ハッシュ値は誰でも計算が可能であるので，不正者は文書管理システムにさえ侵入することができれば，過去のすべてのハッシュを計算し直し，レコードの順序の入れ替えを行うことが可能である．ただし，各レコードそのものは作成者個々人のヒステリシス署名によって守られており，レコードの内容を変更することはできない．不正者があるレコード作成者の秘密鍵と過去の署名履歴を取得できた場合，当該レコード作成者の過去のすべてのヒステリシス署名を計算し直し，署名履歴を改竄することができる．その後，不正者が文書管理システムへの侵入にも成功すると，不正者はレコードの改竄，レコードの順序の入れ替えが可能となる．

(3) ヒステリシス署名方式 B：各レコードのリンキングは文書管理システムのヒステリシス署名により実現されている．また，各レコードの内容そのものは各レコード作成者のディジタル署名により守られている．したがって，文書管理システムの秘密鍵と過去の署名履歴が漏洩した場合，不正者はシステムの過去のすべてのヒステリシス署名を計算し直し，文書中のレコードのリンク情報を改竄することができる．特に，文書本位の署名が行われている場合，「その文書に含まれるすべてのレコードに対する各々の署名」がそのまま「その文書における過去の署名履歴」と一致するので，ヒステリシス署名の再計算は容易である．ただし，各レコードの内容そのものは各レコード作成者のディジタル署名により守られているため，不正者はレコードの内容を変更することはできない．すなわち，不正者はレコードの順序の入れ替えが可能となる．不正者がさらにあるレコード作成者の秘密鍵をも取得することができた場合，当該レコード作成者によって追記されたレコードに関してはその改竄も可能となる．

(4) 多重署名方式：各レコードのリンキングはすべてのユーザとシステムの多重署名によって実現されている．したがって，不正者が文書管理システムの秘密鍵のみを取得したとしても，レコードの改竄や入れ替えは不可能である．同様に，あるレコード作成者 1 人の秘密鍵のみが漏洩しただけでは，レコードの改竄や入れ替えは不可能である．さらに，不正者がシステムの秘密鍵とあるレコード作成者 1 人の秘密鍵のみを取得できたとしても，そのレコード作成者が追記したレコード以降のレコードを作成したユーザのすべての秘密鍵が手に入らなければレコードの改竄や入れ替えは不可能である．

(5) 本方式：各レコードのリンキングはレコードごとに付されているユーザ署名とシステム署名により実現されており，文書管理システムだけではなく，各レコード作成者もレコードのリンキングの主体になっている．すなわち，ヒステリ

シス署名が各ユーザごとに当該ユーザの過去の署名を次々と綴り込んで署名を生成するのに対して，本方式は各文書ごとに複数のユーザおよびシステムの署名を次々と綴り込んで署名が生成されている．したがって，不正者が文書管理システムの秘密鍵のみを取得したとしても，レコードの改竄や入れ替えは不可能であり，基本的に多重署名方式と同様の性質を持つ．

6.3 外部との通信

(1) タイムスタンプ方式：新たにレコードが作成される度に，外部のタイムスタンプサーバにレコードを送信し，リンク情報を得る必要がある．すなわち，管理対象文書が頻繁に追記が起こるような文書である場合には，外部サーバとの通信コストが大きくなる．

(2) ヒステリシス署名方式 A：基本的に外部のサーバとの通信は必要ない．しかし，定期的に署名履歴を外部の第 3 者機関に預けることで，ヒステリシス署名の信頼性を高めることが可能である．

(3) ヒステリシス署名方式 B：ヒステリシス署名方式 A に準ずる．

(4) 多重署名方式：基本的にヒステリシス署名方式 A，B と同様である．

(5) 本方式：基本的にヒステリシス署名方式 A，B および多重署名方式と同様である．

6.4 柔 軟 性

(1) タイムスタンプ方式：タイムスタンプサーバは送られてきた署名付きレコードを機械的に（ハッシュをとって）リンキングするだけであるので，各ユーザは任意に個別の署名アルゴリズムによりレコードに署名を付し，タイムスタンプサーバに送信してかまわない．タイプスタンプサーバ自体は一般的に運用されている公的機関であるため，タイプスタンプサーバ間で（リンクを交差させるなどの）連携をとることは比較的容易であると考えられる．

(2) ヒステリシス署名方式 A：文書管理システムは送られてきた署名付きレコードを機械的に（ハッシュをとって）リンキングするだけである．また，ヒステリシス署名は，署名アルゴリズムそのものに対する制限は基本的には存在せず，過去の署名文を今回の署名に利用しさえすればよい．よって，各ユーザは任意に個別の署名アルゴリズムによりレコードに署名を付し，タイムスタンプサーバに送信してかまわない．異なる署名アルゴリズムによってヒステリシス署名を行っているユーザ間で（履歴を交差させるなどの）連携をとることも可能である．

(3) ヒステリシス署名方式 B：各ユーザからの署名付きレコードに対して，文書管理システムがヒステリシス署名を付加することによってリンキングを行うため，各ユーザの署名アルゴリズム自体に制限はない．また，ヒステリシス署名は，署名アルゴリズムそのものに対する制限は基本的には存在せず，過去の署名文を今回の署名に利用しさえすればよいため，異なる署名アルゴリズムによってヒステリシス署名を行っている文書管理システム間で（履歴を交差させるなどの）連携をとることは可能である．

(4) 多重署名方式：多重署名方式は，文献 9) のアルゴリズムによりレコード間の署名のリンキングを生成しているため，文書管理システムおよびすべてのユーザが文献 9) のアルゴリズムを使用する必要がある．なお，文献 9) のアルゴリズムにより運用されている文書管理システム間であれば（署名を交差させるなどの）連携をとることは可能である．

(5) 本方式：3.3 節および 4.3 節で述べたように，ユーザごと，文書管理システムごとに任意の署名アルゴリズムを用いることが可能であり，どのようなユーザ間，文書管理システム間であっても（署名を交差させるなどの）連携をとることは可能である．

7. ま と め

本論文は，電子化されたライトワンス文書を安全に管理する方式を提案した．本方式は，2 重のディジタル署名を用いてデータの完全性を保証する．具体的には，文書ごとに独立した「文書本位」の署名のリンキングをすべてのユーザと文書管理システムが積極的にリンキングに協力する方式によって行うことにより，文書の改竄困難性が高く，特にライトワンス文書の管理に適したシステムを実現している．本方式は任意の標準的な署名アルゴリズムにより実装可能で，かつ，互いに異なった署名アルゴリズムを用いる複数のユーザが 1 つの文書に対して追記することもでき，運用上の柔軟性を兼ね備えている．また，本論文では，本方式の安全性をさらに向上させる付加的方式について述べるとともに，応用例として電子カルテシステムを示した．そして，本方式と関連方式を比較検討し，本方式の有効性を確認した．

謝辞 5章の電子カルテシステムに関する貴重なご助言をいただいた三菱電機（株）中野初美さんにつつしんで感謝の意を表する．

参 考 文 献

1) Haber, S. and Stornetta, W.S.: How to Time-Stamp a Digital Document, *Journal of Cryptology*, Vol.3, No.2, pp.99–111 (1991).
2) Bayer, D., Haber, S. and Stornetta, W.S.: Improving the Efficiency and Reliability of Digital Time-Stamping, *Sequence II: Methods in Communication, Security and Computer Science*, Springer-Verlag (1993).
3) Buldas, A., Land, P., Lipmaa, H. and Villemson, J.: Time-Stamping with Binary Linking Schemes, *Proc. CRYPTO' 98*, LNCS 1462, pp.486–501 (1999).
4) 高橋 康：電子カルテシステム「MegaOak-NEMR」，NEC 技報，Vol.53, No.9, pp.3–6 (2000).
5) 宮崎一哉，鴨志田昭輝，中川路哲夫：セキュアストレージシステムの開発（1）—電子データの長期保存における原本保証，秘匿，公証技術，第61回情報処理学会全国大会講演論文集，1F-2 (2000).
6) Surety Inc., Digital Notary Service. http://www.surety.com/ （2002年11月20日現在）.
7) NTTデータ，電子文書証明サービスSecure Seal. http://www.ssc.nttdata.co.jp/index2.html （2002年11月20日現在）.
8) 州崎誠一，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, pp.2381–2393 (2002).
9) Mitomi, S. and Miyaji, A.: A General Model of Multisignature Schemes with Message Flexibility, Order Flexibility, and Order Verifiability, *IEICE Trans. Fundamentals*, Vol.E84-A, No.10, pp.2488–2499 (October 2001).
10) Shamir, A.: How to share a secret, *Comm. ACM*, Vol.22, No.11, pp.612–613 (1979).

(平成14年12月10日受付)
(平成15年 6 月 3 日採録)

原田 篤史

平成13年静岡大学情報学部情報科学科卒業．現在，同大学大学院修士課程．情報セキュリティに関する研究に従事．

西垣 正勝（正会員）

平成2年静岡大学工学部光電機械工学科卒業．平成4年同大学大学院修士課程修了．平成7年同大学院博士課程修了．日本学術振興会特別研究員（PD）を経て，平成8年静岡大学情報学部助手．平成11年同講師，平成13年同助教授．博士（工学）．情報セキュリティ，ニューラルネットワーク，回路シミュレーション等に関する研究に従事．

曽我 正和（正会員）

昭和33年京都大学工学部電子工学科卒業．昭和35年同大学大学院修士課程修了．昭和35年～平成8年三菱電機，計算機製作所副所長，情報電子研究所所長を経て平成8年静岡大学情報学部教授，平成11年岩手県立大学ソフトウェア情報学部教授，現在に至る．博士（工学）（東京大学）．汎用計算機，制御用計算機，制御用システムの開発．フォールトトレラントシステム，セキュリティシステムに関する研究に従事．IEEE，電子情報通信学会各会員．

田窪 昭夫（正会員）

昭和41年早稲田大学理工学部電気工学科卒業．昭和43年同大学大学院理工学研究科修士課程修了．同年三菱電機株式会社入社，平成10年静岡大学大学院博士後期課程修了．平成14年東京電機大学情報環境学部教授．博士（工学）．モバイルコンピューティング，ネットワーク・セキュリティ，個人情報保護，情報倫理等に興味を持つ．電気学会，IEEE，ACM各会員．

中村 逸一（正会員）

昭和60年茨城大学工学部卒業，昭和62年同大学大学院修了．同年日本電信電話株式会社入社．LANシステムの研究に従事．平成8年より（株）NTTデータにてセキュリティ技術の研究・開発に従事．現在，同社セキュリティビジネスユニット部長．