追記が付される文書の電子化とその長期的保存のための一考察

藤川真樹[†,††] 八巻睦子[†,†††] 中村雅一[†]

稟議書やカルテ等のように，オリジナル文書に対して追記が付され，かつ最終的な決裁・確定後から数年間にわたる長期的な保存を必要とする文書を電子化する際には，オリジナル文書と追記とを明確に区別できるとともに，その完全性（改ざんされていないこと）と存在性（ある特定の時刻に存在していたということ）を検証できる仕組みが必要である．そこで本論文では，はじめに追記が付される文書を電子化するための方法を提案する．これは，「追記は文書を構成する要素（記載項目）に対して付されるものである」という考え方に基づき，文書を複数の要素（記載項目）に分解するとともに，追記対象の要素に対して追記する要素を時系列に連結したり，文書を構成する要素を新しく追加したりすることによって追記を実現させるものである．その後，ハッシュ値とタイムスタンプを用いて，追記が付された電子文書の完全性と存在性を証明するシステムを提案する．最後に，提案システムの有用性，システムで使用しているハッシュ関数の暗号アルゴリズムが破られる可能性が高くなったときの対処方法，および攻撃者とシステム管理者が結託した場合の脅威とその対策について述べる．

Realization of Electronic Document with Postscript and Considerations in Long Term Preservation

Masaki Fujikawa,[†,††] Mutsuko Yamaki[†,†††] and Masaichi Nakamura[†]

There is a document that postscript is added to an original document and long term preservation is required after final approval. Request for managerial decision and medical report is a typical example. In order to realize an e-document described above, it is necessary to distinguish between original document and postscript accurately and to verify the integrity and existence of it. In this paper, we propose a method that e-document realization with postscript. In our point of view, document is structuerd by element (items mentioned), so we resolved a document into some elements. Postscript is realized that new element will be coupled to the existing element in time series and new element will be add to the document. After that, we propose a model system that can prove the integrity and existence of e-document by using hash value and digital time stamp. In the last place, we describe the following things: (1) Usefulness of our system. (2) Countermeasures when hash function that is used in our system will be broken and (3) when attacker and system administrator will be in collusion.

1. はじめに

近年，電子化されたオリジナル文書の存在性（ある時点に存在していたこと）と完全性（改ざんされていないこと）を証明するための研究が進められており，いくつかの要素技術（電子署名，タイムスタンプ等）[1)~6)]や実装システム[7),8)]が提案されている．このうちタイムスタンプは，ある電子文書 M が特定時刻 T に存在し，M が T 以降改ざんされていないことを検証できる暗号技術であり[9)]，すでに商用サービスとして展開されているものもある[10)~12)]．

さて，文書の中にはオリジナル文書に対して追記（あとから付け足して書き加える文書）が付されるものが存在する．このような文書として，稟議書やカルテ（診療録）等があげられる．稟議書の場合，決裁者によって「但し書き」や「否決理由」が書き加えられることがあり，カルテの場合，医師や医療スタッフによって処置内容が書き込まれたり，文書が添付されることがある[13)]．

またこれらの文書は，最終的な決裁や確定（医師による文書の確認）後からある程度長期的に保存される

† 綜合警備保障株式会社
SOGO KEIBI HOSHO Co., Ltd.
†† 中央大学大学院理工学研究科
Graduate School of Science and Engineering, Chuo University
††† お茶の水女子大学大学院人間文化研究科
Graduate School of Humanities and Sciences, Ochanomizu University

ものである．稟議書は企業等によって保存期間が異なるが，おおむね 1〜3 年間程度である．カルテの場合，医師法第 24 条や歯科医師法第 23 条により最終確定後 5 年間の保存が義務づけられている．当然のことではあるが，保存期間中は当該文書に対する改ざんを防止し，その完全性を維持しなくてはいけない．

このように，追記が付される文書を電子化し，その存在性と完全性をある一定の期間維持するとともに，これらを証明できるようにするためには，どのようにすればよいだろうか．

そこで本論文では，追記が付される文書を電子化するための方法について検討を行い，1 つのモデル・システムを提案する．その後，提案したシステムが当該電子文書の存在性と完全性を維持・証明できることを示すとともに，提案システムにおいて考えられる問題点について議論する．

2. 追記が付される文書の電子化

2.1 準　　備

本論文では，以下に示す文書を取り扱うものとする．

オリジナル文書

起票された直後で追記が付されていない文書を指したもので，これを電子化したものを「オリジナル電子文書」と呼ぶことにする．

追記文書

あとから書き加える文書そのものを指したもので，単に追記と呼ぶこともある．これを電子化したものを「追記電子文書」と呼ぶことにする．

追記付き文書

追記が付された文書を指したもので，これを電子化したものを「追記付き電子文書」と呼ぶことにする．

文書

オリジナル文書または追記付き文書を指したもので，これを電子化したものを単に「電子文書」と呼ぶことにする．

2.2 要素による分解

追記が付される文書を観察すると，追記は「文書全体に対して付されている」というよりも,「文書中の記載項目に対して付されている」といえる．たとえば，稟議書では「稟議する内容」という記載項目に対して追記が付されており，カルテでは「治療歴」,「検査結果」という記載項目[14]に対して追記が付される．

そこで筆者らは,「文書（document）は記載項目という要素（element）の集合体であり，追記は要素に対して付されるものである」と考え，**図 1** に示すように文書 M を要素に分解し，これを Att_i_t $(i = 1, 2, \ldots, n, t = time)$ と定義する．n は文書を構成する要素の総数であり，$time$ は要素の作成時刻を表す．

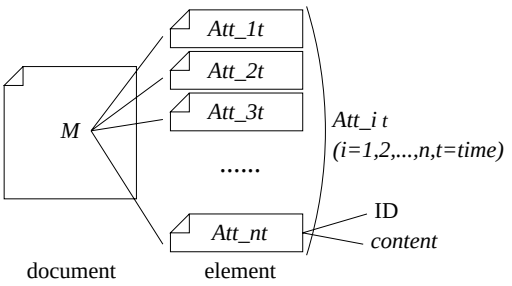

図 1　文書と要素との関係
Fig. 1　Relation between document and element.

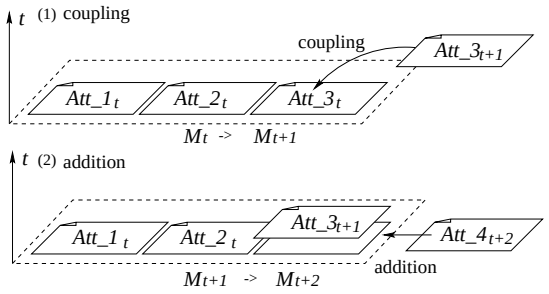

図 2　要素の連結と追加
Fig. 2　Coupling and addition of element.

要素はさらに，記載内容を表す $content$ と ID に分解される．ID は文書を特定するために付与されており，要素から文書への再構成を可能にしている．

たとえば，要素の総数が 3 である電子文書を時刻 t に作成した場合，電子文書 M_t は次式のように表現される．

$$M_t = (Att\_1_t, Att\_2_t, Att\_3_t)$$

2.3 要素の連結と追加による追記の実現

また筆者らは，追記は**図 2** に示すように，

(1) 追記対象の要素に対して新しく生成した要素を時系列に連結（coupling）させること，

(2) 要素の集合体（文書）に対して新たに要素を追加（addition）すること，

により実現できると考えた．(1) は加筆や二重線記入による削除操作が，(2) は追記文書の添付操作が例としてあげられる．

たとえば，前節の電子文書 M_t の要素 Att_3_t に対して時刻 $t+1$ に要素 Att_3_{t+1} を追記（連結）した場合，追記後の電子文書 M_{t+1} は次式のように表現される．

$$M_{t+1} = (Att\_1_t, Att\_2_t, Att\_3_{t/t+1})$$

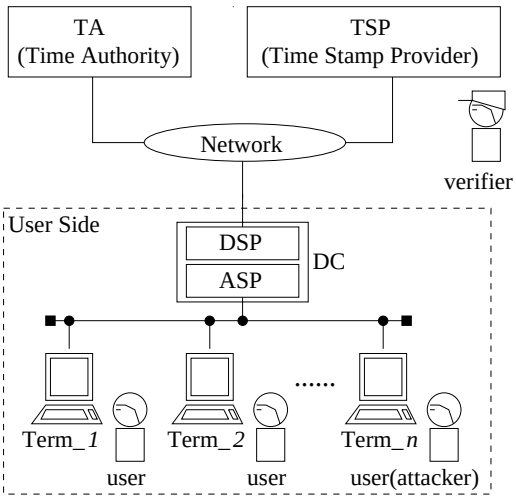

図 3 システム構成
Fig. 3 Composition of the system.

$\because Att\_3_{t/t+1} = (Att\_3_t, Att\_3_{t+1})$

また，電子文書 M_{t+1} に対して時刻 $t+2$ に要素 Att_4_{t+2} を作成し追記（追加）した場合，追記後の電子文書 M_{t+2} は次式のように表現される．

$$M_{t+2} = (Att\_1_t, Att\_2_t, Att\_3_{t/t+1}, Att\_4_{t+2})$$

3. モデル・システムとエンティティ

本論文では，**図 3** に示すようなモデル・システムを提案する．これは，データセンタ DC，端末 Term_j ($j = 1, 2, \ldots, n$)，タイムスタンプ・プロバイダ TSP およびタイム・オーソリティTA によって構成されている．また本システムでは，ユーザ（user），攻撃者（attacker），検証者（verifier）というエンティティが存在する．以下に，システムの構成要素と各エンティティについて述べる．

データセンタ DC

- 端末 Term_j ($j = 1, 2, \ldots, n$) を詐称した端末からのアクセスを防止する「端末認証機能」を備えている．
- Term_j ($j = 1, 2, \ldots, n$) によるアクセスを許可する領域 ASP と，許可しない領域 DSP を持つ．
- 電子文書およびその要素に付される時刻 t が攻撃者によって変更されないようにするために，DSP 内で t を厳正に管理する．なお t は，信頼できる第三者機関である TA（Time Authority）より入手し，サービス要件として設定された精度をつねに満足するように TA の内部時計と同期しているものとする．
- タイムスタンプ・リクエスト TSR を生成して TSP に送信したり，TSP により発行されたタイムスタンプ TS を保存したりする．なお TSR は，TS を取得する TSP が使用しているタイムスタンプ・プロトコルに適合しているものとする．すなわち，DC は「単純なタイムスタンプ・プロトコル」,「リンキング・プロトコル」,「分散プロトコル」等[9]のいずれのタイムスタンプ・プロトコルにも対応可能とする．
- TTP（Trusted Third Party）のような絶対的な信頼性はない装置であるとする．

端末 Term_j ($j = 1, 2, \ldots, n$)

- PIN 入力や指紋照合等によるユーザの認証機能を備えている．
- DC 内の ASP にアクセスすることにより，電子文書の起票，追記および決裁・確定ができるとともに，DC に対して電子文書のタイムスタンプ TS の取得を要求できるものとする．

タイムスタンプ・プロバイダ TSP

- 攻撃者との結託が不可能である信頼された第三者機関とし，DC によって生成・送信された TSR に対して TS を発行し DC に送信する．
- タイムスタンプに付される時刻 t を厳正に管理する機能を備えているものとする．なお t は，信頼できる第三者機関である TA より入手し，サービス要件として設定された精度をつねに満足するように TA の内部時計と同期しているものとする．
- 「単純なタイムスタンプ・プロトコル」,「リンキング・プロトコル」,「分散プロトコル」のいずれかのタイムスタンプ・プロトコル[9]を使用しているものとする．

タイム・オーソリティTA

- 攻撃者との結託が不可能である信頼された第三者機関とし，正確な時刻情報を管理・提供できるものとする．

なお，構成要素間で通信されるデータは，機密性や完全性が確保されているとする．

ユーザ（user）

電子文書の起票者，追記者，決裁・確定権限者，タイムスタンプ取得要求者を総称したもので，端末 Term_j ($j = 1, 2, \ldots, n$) を操作できる人を指す．

攻撃者（attacker）

既存の電子文書に対して，自分の立場が有利にな

るような改ざんを行う人を指す．たとえば，ある患者に対する医療ミスの発覚を恐れて当該患者のカルテを改ざんする医師がこれにあたる．本論文では，ユーザが攻撃者になることを想定している．

検証者（verifier）

攻撃者との結託が不可能である信頼された人のことで，電子文書の存在性と完全性の検証および確認を行う．

4. 電子文書の生成から保存までの処理

本論文では，オリジナル電子文書の生成から追記付き電子文書の長期的保存に至る一連のながれを電子文書の「ライフサイクル」と呼び，これを形成する5つの「プロセス」を定義する．以下に，各プロセスの概要を述べる．

(1) **起票プロセス**
ユーザ（起票者）がオリジナル電子文書の各要素を生成し，これらをまとめてオリジナル電子文書を作成する．

(2) **追記プロセス**
ユーザ（追記者）が既存のオリジナルまたは追記付き電子文書の特定の要素に対して追記を行う．

(3) **決裁・確定プロセス**
ユーザ（決裁・確定権限者）が既存のオリジナルまたは追記付き電子文書に対して決裁・確定を行う．

(4) **タイムスタンプ取得プロセス**
決裁・確定済みのオリジナルまたは追記付き電子文書の存在性と完全性の証明のために，ユーザ（タイムスタンプ取得要求者）がDCに対してタイムスタンプの取得を要求する．DCは当該文書から TSR を生成してTSPに送信し，TSPから TS を取得する．

(5) **保存プロセス**
タイムスタンプの対象となったオリジナルまたは追記付き電子文書と TS を一組にしてDC内に保存する．

ここで，稟議書とカルテを例にとり，ライフサイクルの形成を試みる．

稟議書の場合，起案者による起票後，決裁権限者による追記および決裁が順次行われていき，最終決裁権限者によって追記および決裁が行われたあとに保存されるのが一般的である．このため，**図4**(a)に示すように「起票」後,「追記」→「決裁・確定」というプロセスを何度か繰り返したあとに,「追記」→「決裁・確定」→「タイムスタンプ取得」→「保存」という経路をたどるライフサイクルが形成できる．

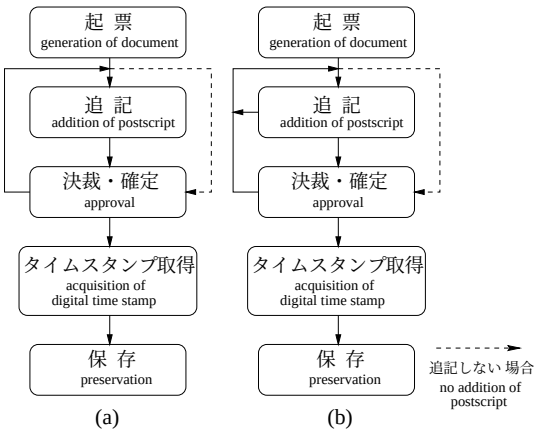

図4 電子文書のライフサイクル
Fig. 4 Life cycle of electronic document.

一方カルテの場合，医師による起票後，医療スタッフによって何度か追記が繰り返されたあと，医師によって追記および確定が行われる．そして，当該文書に対して最終的な確定が行われるまで，追記および確定が繰り返される．このため，図4(b)に示すように「起票」後,「追記」プロセスを何度か繰り返したあとに「決裁・確定」が行われ，その後再び「追記」プロセスを繰り返したあとに,「決裁・確定」→「タイムスタンプ取得」→「保存」経路をたどるライフサイクルが形成される．

以降の節では,「起票」→「追記（要素の連結）」→「追記（要素の追加）」→「決裁・確定」→「タイムスタンプ取得」→「保存」という1つのライフサイクルを例にとり，プロセスごとの処理について述べる．

4.1 起　　票

起票者が任意の端末 Term_j $(j = 1, 2, \ldots, n)$ を操作してDCにアクセスし，ASP内でオリジナル電子文書を起票するものとする．説明のため，電子文書を構成する要素 Att_i_t $(i = 1, 2, \ldots, n, t = time)$ の総数 n を4とする．ここでは，起票者は Att_1_t，Att_2_t，Att_3_t，Att_4_t の各要素を新しく生成するとともに，起票者を特定できる情報 $Info_t$（氏名・所属等を含む）を連結させてオリジナル電子文書 M_t を生成する（要素および電子文書に付される時刻 t はDCが付与）．

$$M_t = (Att\_1_t, Att\_2_t, Att\_3_t, Att\_4_t, Info_t)$$

次にDCは，DSP内でオリジナル電子文書 M_t をもとにハッシュ値 $H(M_t)$ を計算する．なお，ハッシュ値を生成するハッシュ関数Hash()は，Hash (x) = Hash (y) となるような一対の値 (x, y) を求めることが難し

い衝突困難な Hash 関数（collision instractable hash function）とする．

$$H(M_t) = Hash(M_t)$$

生成された $H(M_t)$ を「最新のハッシュ値」と呼ぶことにする．そのあと DC は，ASP 内に M_t を，DSP 内に $H(M_t)$ を保存し，オリジナル電子文書 M_t に対して追記が行われるまで待機する．

4.2 追記（要素の連結）

追記者が任意の端末 Term_j ($j = 1, 2, \ldots, n$) を操作して DC にアクセスし，ASP 内に保存されているオリジナル電子文書 M_t に対して追記（要素の連結）を付すものとする．ここでは，追記者は Att_3_t，Att_4_t に対して追記を付すために Att_3_{t+1}，Att_4_{t+1} をそれぞれ生成し，Att_3_t，Att_4_t に対して以下のように時系列に連結させる．

$$Att\_3_{t/t+1} = (Att\_3_t, Att\_3_{t+1})$$
$$Att\_4_{t/t+1} = (Att\_4_t, Att\_4_{t+1})$$

また，追記者を特定できる情報 $Info_{t+1}$ も同様に $Info_t$ に対して時系列に連結させる．

$$Info_{t/t+1} = (Info_t, Info_{t+1})$$

以上をもとに，追記付き電子文書 M_{t+1} を生成する（要素および電子文書に付される時刻 $t+1$ は DC が付与）．

$$M_{t+1} = (Att\_1_t, Att\_2_t, Att\_3_{t/t+1}, Att\_4_{t/t+1}, Info_{t/t+1})$$

次に DC は，DSP 内で M_{t+1} をもとにハッシュ値 $H(M_{t+1})$ を計算する．

$$H(M_{t+1}) = Hash(M_{t+1})$$

生成された $H(M_{t+1})$ を「最新のハッシュ値」と呼ぶことにする．そのあと DC は，ASP 内の M_t を M_{t+1} に更新して保存するとともに，DSP 内の最新のハッシュ値を $H(M_t)$ から $H(M_{t+1})$ に更新して保存し，追記付き電子文書 M_{t+1} に対して追記が行われるまで待機する．

4.3 追記（要素の追加）

追記者が任意の端末 Term_j ($j = 1, 2, \ldots, n$) を操作して DC にアクセスし，ASP 内に保存されている電子文書 M_{t+1} に対して追記（要素の追加）を付するものとする．ここでは，追記者は M_{t+1} に対して要素 Att_5_{t+2} を新たに追加するとともに，追記者を特定できる情報 $Info_{t+2}$ を $Info_{t/t+1}$ に対して時系列に連結させる．

$$Info_{t/t+1/t+2} = (Info_t, Info_{t+1}, Info_{t+2})$$

以上により，追記付き電子文書 M_{t+2} を生成する（要素および電子文書に付される時刻 $t+2$ は DC が付与）．

$$M_{t+2} = (Att\_1_t, Att\_2_t, Att\_3_{t/t+1}, Att\_4_{t/t+1}, Att\_5_{t+2}, Info_{t/t+1/t+2})$$

次に DC は，DSP 内で M_{t+2} をもとにハッシュ値 $H(M_{t+2})$ を計算する．

$$H(M_{t+2}) = Hash(M_{t+2})$$

生成された $H(M_{t+2})$ を「最新のハッシュ値」と呼ぶことにする．そのあと DC は，ASP 内の M_{t+1} を M_{t+2} に更新して保存するとともに，DSP 内の最新のハッシュ値を $H(M_{t+1})$ から $H(M_{t+2})$ に更新して保存し，追記付き電子文書 M_{t+2} に対して決裁・確定が行われるまで待機する．

4.4 決裁・確定

決裁・確定権限者が任意の端末 Term_j ($j = 1, 2, \ldots, n$) を操作して DC にアクセスし，ASP 内に保存されている追記付き電子文書 M_{t+2} に対して決裁・確定を行うものとする．ここでは，M_{t+2} の各要素と決裁・確定権限者を特定できる情報 Com_{t+3} とを連結させて決裁・確定済の追記電子文書 M_{t+3} を生成する（要素および電子文書に付される時刻 $t+3$ は DC が付与）．

$$M_{t+3} = (Att\_1_t, Att\_2_t, Att\_3_{t/t+1}, Att\_4_{t/t+1}, Att\_5_{t+2}, Info_{t/t+1/t+2}, Com_{t+3})$$

なお，決裁・確定権限者が複数人いる場合には，$Info_{t/t+1/t+2}$ のように決裁・確定時に Com_{t+x} を時系列に連結させるものとする．次に DC は，DSP 内で M_{t+3} をもとにハッシュ値 $H(M_{t+3})$ を計算する．

$$H(M_{t+3}) = Hash(M_{t+3})$$

生成された $H(M_{t+3})$ を「最新のハッシュ値」と呼ぶことにする．そのあと DC は，ASP 内の M_{t+2} を M_{t+3} に更新して保存するとともに，DSP 内の最新のハッシュ値を $H(M_{t+2})$ から $H(M_{t+3})$ に更新して保存し，タイムスタンプ取得要求者から追記付き電子文書 M_{t+3} に対してタイムスタンプ取得要求があるまで待機する．

4.5 タイムスタンプの取得

タイムスタンプ取得要求者が任意の端末 Term_j ($j = 1, 2, \ldots, n$) を操作して DC にアクセスし，ASP 内に保存されている決裁・確定済の電子文書 M_{t+3} について，DC に対しタイムスタンプの取得を要求するものとする．この時点で，ASP 内には M_{t+3} が，DSP 内には最新のハッシュ値 $H(M_{t+3})$ が保存されている．

DC は，この $H(M_{t+3})$ だけをタイムスタンプ・リクエスト TSR 生成のための要素とし，TSP のプロトコルに応じて TSR を生成し TSP に送信する．

TSP では，受信した TSR からタイムスタンプ TS を作成し DC に送信する．そして，TS を受信した DC は TS を DSP 内に保存する．

表 1 エンティティの read 権限
Table 1 Read permission for entity.

	ユーザ	検証者
電子文書 M_{t+3}	allow	allow
最新のハッシュ値 $H(M_{t+3})$	deny	allow
タイムスタンプ TS	deny	allow

allow：read 可　deny：read 不可

4.6 保　　存

DC は，TS を DSP 内に保存した後，ASP 内に保存している M_{t+3} と DSP 内に保存している TS および $H(M_{t+3})$ の 3 つを一組にして保存期間が満了するまでの間 DSP 内に保存する．なお，M_{t+3} の存在性と完全性を保存期間満了まで維持するために，TS のプロトコルに応じて TS の再取得を行う．

たとえば，TSP が TS の発行に電子署名のみを用いている場合には，

- TS の署名検証に必要な TSP の公開鍵証明書の有効期限が切れる前，
- TS に署名を行う TSP の秘密鍵が漏洩あるいは暴露しそうになったとき，

に TS の再取得を行う．ここで再取得する TS は，TSP が新たに用意した TS 署名用秘密鍵によって生成されるものとする．

また，TSP が TS の発行にハッシュ関数のみを用いている場合には，当該ハッシュ関数の暗号アルゴリズムが破られる可能性が高くなったときに TS の再取得を行う．ここで再取得する TS は，TSP が新たに用意したハッシュ関数（暗号学的に十分な安全性を確保していると考えられているもの）によって生成されるものとする．

なお，DSP 内の各データは read のみとし，ユーザと検証者に対して**表 1** に示す制限を設けるものとする．

5. 存在性と完全性の検証

提案するシステムでは，電子文書の存在性と完全性の検証を以下の 2 つの場合に分けて行う．なお，前者は DC が行い，後者は検証者が行うものとする．

5.1 電子文書が ASP 内に保存されている場合

ASP は攻撃者（ユーザ）によるアクセスが可能であるため，既存の電子文書に対する改ざんが予想される．攻撃者は，自分の立場が有利になるように *content* を書き換えたり，要素そのものを消去できるものと仮定する．前章で示した電子文書のうち，以下のものが改ざんの対象となる．

- オリジナル電子文書 M_t
- 追記付き電子文書 M_{t+1}，M_{t+2}
- 決裁・確定済の追記付き電子文書 M_{t+3}

本システムでは，DC が以下に示すアルゴリズムを用いて電子文書の存在性と完全性を検証する．なおこの処理は，電子文書に対して追記，決裁・確定およびタイムスタンプの取得を行う直前にも実行する．

(1) ASP に保存されている電子文書 M_{t+x} からハッシュ値 $H(M_{t+x})'$ を計算．
(2) DSP に保存されている M_{t+x} の最新のハッシュ値 $H(M_{t+x})$ と $H(M_{t+x})'$ を比較し，一致ならば OK を，不一致ならば NG を出力する．
(3) 出力が OK ならば，プロセス最終実施時刻 $t+x$ に電子文書が存在し，かつ当該文書に対して改ざんがなかったと判定し，出力が NG ならば，電子文書に対して改ざんがあったと判定する．

5.2 電子文書が DSP 内に保存されている場合

DSP はユーザによるアクセスが不可能であるため，攻撃者による改ざんは困難であるといえる．しかし，保存している電子文書の存在性と完全性を証明する必要がある．本システムでは，検証者が以下に示すアルゴリズムを用いて電子文書の存在性と完全性を検証する．

(1) 検証者は，DSP に保存されている電子文書 M_{t+x}，最新のハッシュ値 $H(M_{t+x})$ およびタイムスタンプ TS を取り出す．
(2) 検証者は，TS を発行した TSP から TS の検証に必要なデータを集め，タイムスタンプ・プロトコルに応じて TS の検証を行う．
(3) 検証が成功すれば，タイムスタンプ取得時刻に電子文書が存在し，かつ当該文書に対して改ざんがなかったと判定し，検証が失敗すれば，電子文書に対して改ざんがあったと判定する．

6. 議　　論

6.1 システムの有用性

本システムでは，タイムスタンプ取得前の電子文書の場合，ユーザによって電子文書の生成，追記および決裁・確定が行われるごとに電子文書の最新のハッシュ値を生成し，これを更新・保存することで電子文書の完全性と存在性を証明可能にしているが，「タイムスタンプ」や「ヒステリシス署名」[17),18)]を用いても同様のシステムを構築することができる．

「タイムスタンプ」方式では，電子文書の生成，追記および決裁・確定を行ったユーザが，当該電子文書に対するタイムスタンプを TSP から取得して保存するようにする．これにより，いつ，どのユーザが生成，追記，決裁・確定を行ったかが TSP によって証明さ

表 2 1つの電子文書に対するメモリ量
Table 2 Quantity of memory for one e-document.

	提案システム	タイムスタンプ	ヒステリシス署名
メモリ量	$Q+h$	$Q+ts(1+n+m)$	$Q+sig(1+n+m)$

れ，かつ提案手法よりもシンプルなシステムを構築することができる．

また「ヒステリシス署名」方式では，電子文書の生成，追記および決裁・確定を行ったユーザが，Chaining Signature を用いて当該電子文書に署名をし，署名履歴を保存するようにする．これにより，どのユーザが生成，追記，決裁・確定を行ったかを明確にすることができ，さらに電子署名の非生成も証明できるため，電子文書に対するプロセスの非実行を証明できる．

しかしこれらの手法は，ユーザがある電子文書に対して追記または決裁・確定を行う前に，ユーザあるいは DC が，(1) 最後の追記者または決裁・確定者からタイムスタンプ（電子署名）を入手し，(2) 当該電子文書の検証を行う．という 2 つのステップを踏む必要がある．これに対し，提案システムでは検証データ（最新のハッシュ値）を DC に保存しているため，DC が (1) 当該電子文書の検証を行う．という 1 つのステップを踏むだけでよいため効率的である．

また本システムでは，上述の 2 つの手法に比べて 1 つの電子文書に対するメモリ量を抑えることができる．試みに，起票後の追記回数が n 回，決裁・確定回数が m 回である，メモリ量 Q の電子文書を想定し，それぞれの手法におけるメモリ量の概算を**表 2** に示す．

「タイムスタンプ」方式では，ユーザがプロセス実行ごとにタイムスタンプを取得するため，1 つの電子文書に対してシステム全体として $1+n+m$ 個のタイムスタンプを保存する．ここでタイムスタンプ 1 個のメモリ量を ts とした場合，当該電子文書に対するメモリ量は $Q+ts(1+n+m)$ となる．

「ヒステリシス署名」方式では，ユーザがプロセス実行ごとに電子署名（1 つ前の署名結果のハッシュ値を含む）を生成するため，1 つの電子文書に対してシステム全体として $1+n+m$ 個の電子署名を保存する．ここで電子署名 1 個のメモリ量を sig とした場合，当該電子文書に対するメモリ量は $Q+sig(1+n+m)$ となる．

提案手法では，プロセス実行ごとに最新のハッシュ値を更新・保存するため，追記，決裁・確定回数に関係なく 1 つの電子文書に対してシステム全体として 1 個のハッシュ値を保存する．ここでこのハッシュ値のメモリ量を h とした場合，当該電子文書に対するメモリ量は $Q+h$ となり，他の 2 つの手法に比べて 1 つの電子文書に対するメモリ量を抑えることができる．

6.2 ハッシュ関数のリニューアル

本システムにおいて，電子文書の最新のハッシュ値生成に使用しているハッシュ関数 Hash() の暗号アルゴリズムが破られる（一方向性が保てなくなる）可能性が高くなったり，Hash() の後継バージョンが策定されて，そのバージョンアップの移行期間に入ったりした場合，Hash() をリニューアルする必要がある．このため本システムでは，ハッシュ関数リニューアル時に以下に示すアルゴリズムを実行する．なお前提として，ハッシュ関数のリニューアル時に，6.3 節で述べるような DC のシステム管理者と攻撃者との結託はないものとする．

(1) DC 内 ASP に対するユーザからのアクセスをいったん停止する．
(2) ASP 内に保存している電子文書 M_1 に対して，5.1 節で示したアルゴリズムと Hash() を適用し，改ざんの有無を検証する．ここで改ざんを検出した場合は，M_1 に対する以降の処理を中止するとともに，改ざんを検出したことをアラーム等を用いて通知する．
(3) リニューアル用ハッシュ関数 SHash() を DC に組み込む．
(4) SHash() を用いて，ASP 内に保存している電子文書 M_1，および TS 取得後 DSP 内に保存している電子文書 M_2 それぞれについて「最新のハッシュ値」$SH(M_1)$，$SH(M_2)$ を生成する．
(5) 生成した 2 つのハッシュ値のうち，$SH(M_1)$ は DSP 内に保存する．また $SH(M_2)$ は TSR に変換し，TSP に送信して TS を再取得した後，M_2，$SH(M_2)$ とともに DSP 内に保存する．
(6) DC 内 ASP に対するユーザからのアクセスを再開する．

なお，SHash() をネットワーク経由で DC に組み込む際には，低コストで効率良く実施することが望ましい．最近では，このような暗号アルゴリズムのリニューアルに関する方法がいくつか提案されている[15),16)]が，本システムに対してもネットワークを介して SHash() を組み込む方法を検討する必要がある．

6.3 システム管理者との結託による改ざん攻撃とその対策

本節では，DCを管理するシステム管理者を想定し，攻撃者とシステム管理者との結託によって可能となる電子文書の改ざん攻撃について考察する．本論文では，特に以下に示す攻撃を想定し，その対策について論じる．

攻撃内容：

攻撃者は，ASP またはDSP 内にすでに保存されている電子文書を改ざん（ある時刻に追記された文書の書き換えや追記文書そのものの消去）を実行するために，DCを管理しているシステム管理者と結託する．そしてシステム管理者に対し，

(1) DSP 内に保存されている最新のハッシュ値の消去を依頼した後，ASP 内に保存されている電子文書を改ざんする．その後改ざん後の電子文書から最新のハッシュ値を生成し，これを DSP 内に保存させることで攻撃を成功させる．

(2) DSP 内に保存されている電子文書の改ざんを直接依頼する．依頼されたシステム管理者は，DSP 内に保存されているタイムスタンプと最新のハッシュ値を消去した後電子文書を改ざんし，改ざん後の電子文書から最新のハッシュ値を生成する．その後，当該文書に対するタイムスタンプを取得し直し，DSP 内に改ざん後の電子文書，最新のハッシュ値およびタイムスタンプを保存することで攻撃を成功させる．

前提条件として，システム管理者は DSP 内に保存されている TS の消去，最新のハッシュ値の生成と消去，および M_{t+x} 内の *content* の書き換えと要素の消去ができるものとする．この条件のもとで想定される具体的攻撃方法を以下に示す．

(1) の場合：ASP 内に保存されている，時刻 $t+x$ に作成された電子文書 M_{t+x} を改ざんするために，システム管理者が DSP 内の最新のハッシュ値 $H(M_{t+x})$ を消去した後，攻撃者が M_{t+x} を改ざんし M'_{t+x} を生成する．その後 ASP 内に M'_{t+x} を，DSP 内に改ざん後の最新のハッシュ値 $H(M'_{t+x})$ を保存する．

(2) の場合：DSP 内に保存されている，時刻 $t+x$ に作成された電子文書 M_{t+x} を改ざんするために，システム管理者が DSP 内に保存されている最新のハッシュ値 $H(M_{t+x})$ およびタイムスタンプ TS を消去した後，M_{t+x} を改ざんし，M'_{t+x} と最新のハッシュ値 $H(M'_{t+x})$ を生成する．その後 M'_{t+x} に対するタイムスタンプ TS' を取得し直し，M'_{t+x}，$H(M'_{t+x})$ とともに DSP 内に保存する．

これらの攻撃はいずれも，(1) DSP 内に保存されているデータの改ざんが可能であり，電子文書が改ざんされたことを検証者が検出できないこと，(2) 電子文書および要素の生成時刻とプロセスの実行時刻との相関関係を検証者が証明できないこと，が原因であると考えられる．

上述の問題点を解決するために，本論文では DSP 内に Schneier らが提案した方式[19]を用いた検証ログを保存させる．前提条件として，DSP はこの検証ログを長期的に保存できるとともに，これを保存するための十分なメモリ容量を備えているものとする．また，TTP として新たに ARCHIVE を設ける．ARCHIVE の機能と役割を以下に示す．

- 攻撃者との結託が不可能である信頼された第三者機関とし，DC から送信されたデータを厳正に保管する機能を持つ．
- 検証者の要求に応じて DC から完全な検証ログを取り寄せるとともに，この検証ログの解読処理および完全性の検証処理を行うものとする．そして検証ログの解読結果および完全性の検証結果のすべてを検証者のみに公開する．

ここで，以降で使用する標記について説明する．

ID_x：エンティティ x を一意に特定するための識別子．

$PKE_{PK_x}(K)$：エンティティ x の公開鍵で K を暗号化する．

$SIGN_{SK_x}(Z)$：エンティティ x の秘密鍵で Z に署名する．

$E_K(X)$：共通鍵 K で X を暗号化する．

$MAC_K(X)$：共通鍵 K で X のメッセージ認証子を作成する．

Schneier らは，**図 5** に示すログ記録アルゴリズムを用いてデータをログに記録させている．D_j はログの j 番目に記録されるデータを，W_j は D_j の記録タイプを，A_j は j 番目の記録データに対する認証鍵を表す．D_j をログに記録する際には，装置 U は以下の処理を実行する（A_0 は事前に TTP である T に寄託されているものとする）．

(1) 共通鍵 $K_j(= H(W_j, A_j))$ を生成．

(2) K_j により D_j を暗号化（$E_{K_j}(D_j)$）した直後に K_j を完全に消去．

(3) ハッシュチェーン $Y_j = H(Y_{j-1}, E_{K_j}(D_j), W_j)$

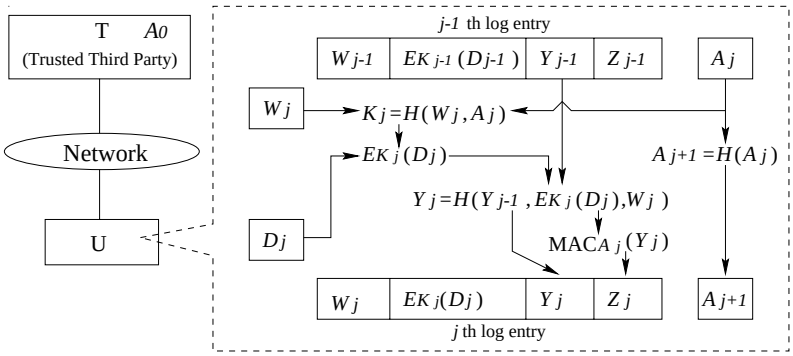

図 5 Secure Audit Log の生成
Fig. 5 Creation of Secure Audit Log.

を生成.

(4) メッセージ認証子 $Z_j = MAC_{A_j}(Y_j)$ を生成.

(5) j 番目のログ $L_j = (W_j, E_{K_j}(D_j), Y_j, Z_j)$ を生成.

(6) 認証鍵 $A_{j+1} = H(A_j)$ を生成後，A_j を完全に消去.

これにより，もし悪意を持った人が U に保存されているすべてのログと A_{j+1} を取得しても，A_{j+1} 以前の鍵 $A_0 \sim A_j$ を導出できないためログを解読したり改ざんしたりすることはできない.

さて，Schneier らの手法を本システムに適用させる場合，T を ARCHIVE に，U を DC に置き換える．そして 1 つの電子文書に対し 1 つの検証ログファイルを生成して 1 対 1 対応にするとともに，D_j には，

- 生成，追記および決裁・確定プロセス実行時には「プロセス実行時刻」とその際に生成・更新される電子文書の「最新のハッシュ値」，
- タイムスタンプ取得プロセス実行時には「プロセス実行時刻」とその際に取得した電子文書の「タイムスタンプ」，
- 保存プロセス実行時には「プロセス実行時刻」と「電子文書」，および電子文書の「最新のハッシュ値」と「タイムスタンプ」，

を含ませ，W_j には実行プロセス名（たとえば生成であれば “**Generation**”，追記であれば “**Addition of P.S.**” 等のように文字列で表記）を含ませる．そして各プロセス実行時に上述のログ記録アルゴリズムを実行させ，検証ログに順次記録させていくようにする．これにより，攻撃者が ASP 内の電子文書を改ざんしたり，システム管理者が DSP 内の電子文書を改ざんしたりしても，システム管理者はログに記録されている上記データを変更できないため，ASP 内の電子文書から再度ハッシュ値を生成してログに記録されているハッシュ値と比較したり，ログに記録されているタイムスタンプを検証したりすることで改ざんを検出することができる.

なお,「検証ログ生成開始時」,「ログ記録終了時」および「ログ検証時」の処理を以下に述べる.

検証ログ生成開始は，電子文書を生成する前に行う．前提として，ARCHIVE は DC から送られてくるデータを確実に受け取るものとする．そして DC は ARCHIVE の公開鍵を持っており，かつ DC の公開鍵証明書は ARCHIVE から発行されているものとする.

(1) DC は，
- ランダムに生成したセッション鍵 K_0
- 現在時刻情報 d
- DC がタイムアウトする時刻情報 $d+$
- ログの識別子 ID_{log}
- DC の公開鍵証明書 C_{DC}
- ランダムに生成した認証鍵 A_0
- プロトコルステップ識別子 p
- $X_0 = (p, d, C_{DC}, A_0)$

より $M_0 = (p, ID_{DC}, PKE_{PK_{ARCHIVE}}(K_0), E_{K_0}(X_0, SIGN_{SK_{DC}}(X_0)))$ を生成し，ARCHIVE に送信する.

(2) DC は，$W_0 =$ **LogfileInitializationType**, $D_0 = (d, d+, ID_{log}, M_0)$ をもとに前述のログ記録アルゴリズムを用いて最初のログ L_0 を生成する．なお DC は A_1 生成後，A_0 を保存しないようにしなければならない.

(3) ARCHIVE は DC から送られてきたデータ M_0 を DC の公開鍵証明書を用いて検証し，合格すれば以下のデータを生成し DC に送信する.

$$M_1 = (p, ID_{ARCHIVE}, PKE_{PK_{DC}}(K_1), E_{K_1}(X_1, SIGN_{SK_{ARCHIVE}}(X_1)))$$

ここで $X_1 = (p, ID_{log}, H(X_0))$，$K_1$ はランダムに生成したセッション鍵である．

(4) DCは M_1 を受け取り，それを検証する．もし検証に合格したならば，W_1=**ResponseMessageType**，$D_1 = M_1$ をもとに新しく L_1 を生成する．もし M_1 をタイムアウト時刻 $d+$ 以内に受け取らなかったり，M_1 の検証が不合格であったりした場合には，W_1=**AbnormalCloseType**，D_1 = (現在の時刻情報とログ記録終了理由) をもとに新しく L_1 を生成し，ログ記録を終了する．

ログ記録終了は，ログ記録を完全に終了させるときに行う．DCは最終記録データ D_f = (現在の時刻情報)，W_f=**NormalCloseMessage** を f 番目の記録とし，L_f を生成するとともに，L_f 生成時に使用した A_f と K_f を完全に消去する．

ログ検証は，必要に応じてそのつど行う．ARCHIVEは検証者の検証要求に応じてDCからすべての検証ログを受け取る．そしてARCHIVEが所持している A_0 をもとにハッシュチェーンと最後のメッセージ認証子を用いてログの完全性を検証する．ARCHIVEはまた，ログ記録アルゴリズムで用いられている共通鍵をすべて導出し，検証ログ全体を解読する．そしてログの完全性検証結果と解読したログを検証者にすべて提示し，検証者はそれらから検証ログの有効性，および電子文書の完全性と存在性を最終的に判断する．

7. まとめ

本論文では，はじめに稟議書やカルテのようにオリジナル文書に対して追記が付される文書を電子化するための方法について述べた．筆者らは，追記は文書全体に対して行われるのではなく，文書を構成している要素（記載項目）について行われているという考え方をもとに，文書を要素に分割するとともに，要素を時系列に連結あるいは追加することにより追記を実現できることを述べた．

続いて，電子文書の生成から保存までのライフサイクルを形成する「起票」・「追記」・「決裁・確定」・「タイムスタンプ取得」・「保存」という各プロセスを定義し，それぞれについて具体的例を交えて説明を加えた．

その後，電子文書の存在性と完全性を検証する方法について，電子文書がASPに保存されている場合とDSPに保存されている場合に分けて述べた．

最後に，提案したシステムについて議論を行い，(1) システムの有用性，(2) システムで使用しているハッシュ関数の暗号アルゴリズムが破られる可能性が高くなったときの対処方法，(3) 攻撃者とシステム管理者が結託した場合における電子文書の改ざんを検出するための手法について検討を加えた．

謝辞 ご多忙のところ，懇切丁寧かつ多くの有益なコメントをいただいた査読者の方に感謝いたします．

参考文献

1) 岡本龍明，山本博資：現代暗号，産業図書 (1997).
2) 辻井重男，笠原正雄：暗号と情報セキュリティ，昭晃堂 (1990).
3) Massias, H. and Quisquater, J.J.: Time and Cryptography, Belgian Project TIMESEC Technical Report WP1 (1997).
4) Adams, C., Cain, P., Pinkas, D. and Zuccherato, R.: *Internet X.509 Public Key Infrastructure Time Stamp Protocols* (1999).
5) Fábrica Nacional de Moneda y Timbre: PKITS Overview Final Report. http://www.cordis.lu/infosec/src/winners.htm (2001.9.17 現在).
6) ETSI ES 201 733 Electronic Signature Formats. http://portal.etsi.org/sec/el-sign.asp (2001.9.17 現在).
7) 国分明男，谷内田益義，山口雅浩：原本性保証電子保存システムの開発. http://www.nmda.or.jp/nmda/ipa/gen/ipa-gen.html (2001.9.17 現在).
8) 宮崎一哉，鴨志田昭輝，中川路哲男：セキュアストレージシステムの開発（1），第61回情報処理学会全国大会講演論文集，1F-2 (2000).
9) 宇根正志，松浦幹太，田倉 昭：デジタルタイムスタンプ技術の現状と課題，金融研究第19巻別冊第1号，pp.105–154, 日本銀行金融研究所.
10) NTT データ電子文書証明サービス（Secure Seal）．http://210.144.76.11/index2.html (2001.9.17 現在).
11) Surety.com Digital Notary Service. http://www.surety.com/index-nn.html (2001.9.17 現在).
12) BML Co., Ltd：電子カルテにタイムスタンプを発行するサービス．http://www.bml.co.jp/ (2001.9.17 現在).
13) カルテ記載の統一基準（医療法人寺西報恩会長吉総合病院）．http://members.tripod.co.jp/kohoo_ngh/s02karute.htm (2001.9.17 現在).
14) 電子保存と電子カルテ（藤田保健衛生大学医学部放射線医学教室）．http://radiology.fujita-hu.ac.jp/radiology/informatics/ (2001.9.17 現在).
15) 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.41, No.8, pp.2101–2109 (2000).
16) 栃窪孝也，岡田光司，遠藤直樹，岡本栄司：リニューアル可能な暗号認証システムの検討，情

報処理学会論文誌，Vol.41, No.8, pp.2121–2128 (2000).
17) 松本 勉，岩村 充，佐々木良一，松本 武：暗号ブレイク対応電子署名アリバイ実現機構（その 1)—コンセプトと概要，情報処理学会研究報告 2000-CSEC-8, pp.13–17 (2000).
18) 洲崎誠一，宮崎邦彦，宝木和夫，松本 勉：暗号ブレイク対応電子署名アリバイ実現機構（その 2)—詳細方式，情報処理学会研究報告 2000-CSEC-8, pp.19–24 (2000).
19) Schneier, B. and Kelsey, J.: Cryptographic Support for Secure Logs on Untrusted Machines, *Proc. 7th USENIX Security Symposium*, pp.53–62, USENIX Press (1998).

(平成 13 年 4 月 10 日受付)
(平成 13 年 9 月 12 日採録)

藤川 真樹（正会員）

1996 年徳島大学工学部知能情報工学科卒業．1998 年同大学大学院工学研究科知能情報工学専攻博士前期課程修了．同年綜合警備保障株式会社入社．現在，同社技術研究所にて情報セキュリティに関する研究に従事するかたわら，中央大学大学院理工学研究科博士後期課程に在籍．電子情報通信学会，日本医療情報学会各会員．

八巻 睦子（学生会員）

1998 年お茶の水女子大学大学院家政学研究科博士前期課程修了．1999 年綜合警備保障株式会社入社．以来，同社技術研究所にて高齢者介護における情報通信システムの適用可能性に関する研究に従事するかたわら，お茶の水女子大学大学院博士後期課程に在籍．日本老年社会科学会，日本福祉介護情報学会各会員．

中村 雅一

1976 年慶應義塾大学理工学部電気工学科卒業．同年綜合警備保障株式会社入社．現在，同社開発本部にて情報セキュリティ・システムの開発に従事．