経済産業省委託調査

平成１４年度ＥＣ技術基盤の相互運用性に関する調査研究事業

（電子署名生成・検証システムのセキュリティ環境の国際標準化等の調査）

タイムスタンプサービスの利用ガイドライン

平成15年3月

電子商取引推進協議会
財団法人日本情報処理開発協会
電子商取引推進センター

この報告書は、平成１４年度受託事業として（財）日本情報処理開発協会電子商取引推進センターが経済産業省から委託を受けて、電子商取引推進協議会（ECOM）の協力を得て実施した「平成１４年度ＥＣ技術基盤の相互運用性に関する調査研究事業（電子署名生成・検証システムのセキュリティ環境の国際標準化等の調査）」の成果を取りまとめたものです。

はじめに

タイムスタンプは、その電子文書がタイムスタンプ生成時刻以前に確かに存在した事を証明（存在証明）すると共に、その当時の状態を保持していることを証明（完全性証明）することができる。電子署名文書の長期保存にタイムスタンプは不可欠であり、タイムスタンプサービスが高い信頼を得て長期にわたりそのサービスを継続し、社会インフラとして広く認められることは、非常に重要である。

現在、ワールドワイドに複数の時刻源が存在しており、既に、幾つかのタイムスタンプサービスが提供され始めた。

本ガイドラインでは、タイムスタンプサービスの必要性、利用例、方式別特徴、及び利用上の留意点をまとめた。

本ガイドラインが、日本企業・各機関の方々にとって、タイムスタンプの理解の一助になれば幸いである。

平成 15 年 3 月

財団法人日本情報処理開発協会
電子商取引推進センター
電子商取引推進協議会

目　次

1. タイムスタンプの必要性

本章では、タイムスタンプが必要となった技術的な背景と、タイムスタンプが必要とされる実際の利用シーンなどについて述べる。

1.1 タイムスタンプとは

タイムスタンプとは、電子文書等のデータに対してタイムスタンプ生成時にその電子文書が存在したことを証明すると共に、その当時の状態を保持したことを証明できるようにするために生成されたデジタルデータのことである。タイムスタンプには時刻情報が含まれており特定の日時における文書等のデータが存在したことの証明となっているが、この時刻をいかに正確・安全にタイムスタンプに反映するかは様々な方式がある。本書では技術面や運用面など何らかの方法により、タイムスタンプが時刻に対して信頼性を持つものを対象としている。

1.2 技術的な背景

タイムスタンプが必要とされる技術的な背景となる諸問題について述べる。

1.2.1 コンピュータの時刻に関する問題

インターネット上の Web サイトを構築しているコンピュータ（サーバ）や、企業間取引などに用いられているコンピュータは、各々独自の内部クロックにより時刻を生成させているため下記の問題がある。

- 複数のコンピュータ同士の時刻は、ずれている場合がある。
- あるコンピュータの時刻は、標準時とずれている場合がある。
- コンピュータの時刻はシステム管理者などにより任意の時刻（日付）を自由に設定することができる。

日付を自由に設定できる場合、このコンピュータ上で実施されたデジタル署名については、署名を実施することのできる者であれば任意の日付（時刻）のデジタル署名を自由に生成できてしまうことを意味する。

1.2.2 電子文書の特性に関する問題

紙に代わり電子的な状態を「原本」とした場合、下記の問題がある。

- 電子文書には物理的な実態が存在しないため、物質特性（紙の老朽化、印刷のかすれなど）から文書の生成年月を推測することはできない。そのため、あたかも過去に存在したかのように全く新たに電子文書を作成（偽造）してもそれを判別する方法は無い。
- 電子文書を複製した場合、複製は原本と区別できない。そのため、CD-R などの一回限りの書き込みメディア等を利用した保管を原本として管理していた場合でも、何らかの手段で電子文書を入手した何者かが、全く新たに「原本メディア」を作成（偽造）しても、それを容易には判断できない場合がある。

1.2.3 PKIに関する問題

PKI によるデジタル署名を利用することで、電子文書の「否認防止」「完全性」は確保できる。しかし PKI は暗号技術によりセキュリティの確保を行っているため、デジタル署名や公開鍵証明書には将来コンピュータの処理速度向上やアルゴリズムの脆弱性露見を見越した有効期限があり、この有効期限が切れてしまった場合、PKI による「否認防止」や「完全性」の効果は無くなってしまう。この有効期限の多くは１～５年に設定されており、デジタル社会で求められている文書の保存年数に満たない場合が多い。

また何らかの理由により鍵の危殆化が懸念された場合は、有効期限前であっても公開鍵証明書の有効性が失われてしまい、デジタル署名の有効性が失われてしまう可能性もある。

1.3 タイムスタンプの役割

1.3.1 タイムスタンプで証明できるもの

タイムスタンプを用いることで、下記の二つが証明できる。

・電子文書の存在証明

タイムスタンプが生成された時刻以前に、タイムスタンプが施された電子文書が確かに存在したことを証明するもの。

・電子文書の完全性証明

タイムスタンプを施された電子文書が、その当時の状態を保持していることを証明するもの。

1.3.2 タイムスタンプの仕組み

タイムスタンプは次の手順で生成される（図 1-1）。

① 要求者が証明したい電子文書のハッシュ値を生成する
② それをタイムスタンプ発行者（ＴＳＡ）に送る
③ タイムスタンプ発行者（ＴＳＡ）は時刻配信者からの時刻を定期的あるいは随時に受けて正確な時刻を保持している。
④ その時刻を含み何らかの関連付けを安全に実施する
⑤ その結果（データ）を要求者に返信する。

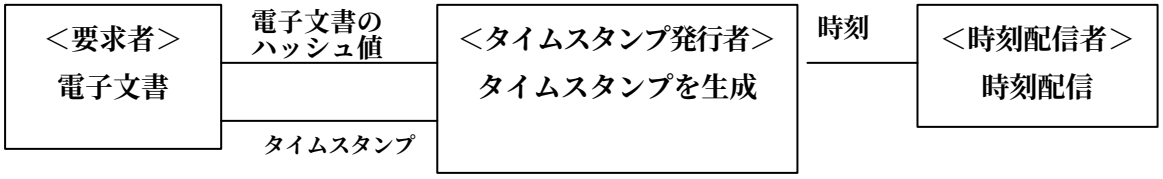


図 1-1 タイムスタンプの生成

電子文書がその当時に確かに存在し、その当時の内容と相違ないことを確認するためには、タイムスタンプの検証を行うことで判断できる。
タイムスタンプの検証方法は、方式により様々あり詳細は次章で述べる。

1.3.3 タイムスタンプの信頼性について

タイムスタンプは、保証する時刻の精度や、時刻配信者、タイムスタンプ発行者のいずれかにおいて不正を発生させない工夫や、発生したことを検出できる仕掛けの有り無しに応じて信頼性の評価ができる。タイムスタンプ発行者の信頼性については下記の観点で評価できる。

- タイムスタンプ発行システムに関する運用規定等がある。
- 運用規定の遵守を監視する監査等の体制が整っている。
- 耐タンパ装置などを用いてログ等の改ざんを防いでいる。
- 標準時刻と正確に同期を取り誤差を修正する仕掛けがある。

1.3.4 利用シーン毎の効果

下記の事例は、電子化に伴うセキュリティ上の問題がタイムスタンプの存在で解決される例である。

表 1-1 利用シーン毎のタイムスタンプの効果

	電子化に際しての問題・課題	期待されるタイムスタンプの効果
電子入札 電子申請	電子化された状態の文書を「原本」とするため、作成（発生）日付の特定ができなくなる。	入札や申請が行われた日時の特定や内容の完全性を確保する。
オンライントレード	注文などの電子商取引において事象の前後関係や、オークション価格など時刻により変動するものを扱う場合、単なる時刻の設定ミスによる誤りや、空取引など様々な問題がある。	トラブル発生時に事象（注文）の存在や取引の内容や価格などが改ざんされていないことを証明できる。
電子カルテ	様々な医療記録（カルテ、写真、電子機器による診断データ、など）が電子化され、これら記録の電子的な保管がなされた時に、過去に遡って内容の改ざんを行われても分からない。	カルテの改ざんなどを防ぐことで医療機関の業務の信頼性を証明することができる。
特許 知的財産	企業などによる研究開発業務の成果として生成された電子的な情報は、それ自体が特定の日時に存在していたことを証明できなければ、発明の先進性証明の助けにならない。	特定の日時に発明が存在していたことを検証し、特許に関連した係争の場合に証拠として活用できる。
郵便	電子メールは、手紙としてみた場合、存在の証明と送付の確実性を保証しない。	タイムスタンプは「消印」に相当する。

2．タイムスタンプの利用例

2.1　タイムスタンプの利用モデル

タイムスタンプの典型的な利用モデルを図 2-1 に示す。利用モデルに登場するエンティティは次のとおりである。

(1) 要求者：
タイムスタンプクライアントを用いてタイムスタンプ発行者にタイムスタンプを要求し、取得する。
(2) 検証者：
タイムスタンプクライアントを用いてタイムスタンプを検証する。あるいは、タイムスンタプクライアントを用いてタイムスタンプ発行者にタイムスタンプの検証を要求し、検証結果を取得する。
(3) タイムスタンプ発行者（TSA）：
タイムスタンプサーバを用いてタイムスタンプを発行する。タイムスタンプの検証要求を受け、検証結果を発行する場合もある。また、発行した複数のタイムスタンプから生成したハッシュ値等の証拠情報を新聞や定期刊行物などの公開メディアに掲載する場合もある。
(4) 時刻配信者（TA）：
専用の装置やソフトを用いて、時刻配信（タイムスタンプサーバに標準時刻を配信する）、時刻監査（タイムスタンプサーバを特定し、そのクロックの標準時刻との差異を計測する）、時刻認証（タイムスタンプサーバを特定、そのクロックの標準時刻との差異を計測し、差異が存在した場合にそのクロックを調整して標準時刻と同期をとる）を実施する。時刻監査を行った事実やその結果、あるいは時刻認証を行った事実や内容を証明する証明書（「時刻監査証明書」と呼ぶこととする）を発行することもある。
(5) 国家標準時配信者（NTA）：
国家の標準時刻を生成し、TA に配信する。TSA が NTA から直接配信を受けることも可能である。
(6) 監査人：
タイムスタンプ発行者のログや証拠情報などを検証することにより、信頼に足るタイムスタンプ発行がなされているか否かを検証する。
(7) 認証局：
デジタル署名ベースのタイムスタンプや時刻監査証明書を用いる場合、タイムスタンプ発行者や時刻配信者に公開鍵証明書を発行する。
(8) リポジトリ：
デジタル署名ベースのタイムスタンプを検証する場合に必要となる公開鍵証明書や証明書失効情報を発行する。証明書検証のために OCSP レスポンダなどの検証サーバを利用する場合もある。

(9) 署名ポリシ発行者：

電子署名文書にタイムスタンプを添付する際に、署名ポリシによってタイムスタンプを有効であると判断するための要件を指定することがある。その場合、検証者は署名ポリシに従ってタイムスタンプの有効性を検証する必要がある。署名ポリシ発行者は、署名ポリシの参照情報（OID など）と署名ポリシの内容を登録／管理し、電子署名文書内に示された署名ポリシの参照情報をキーとした問い合わせに応え、署名ポリシ本体情報を発行する。

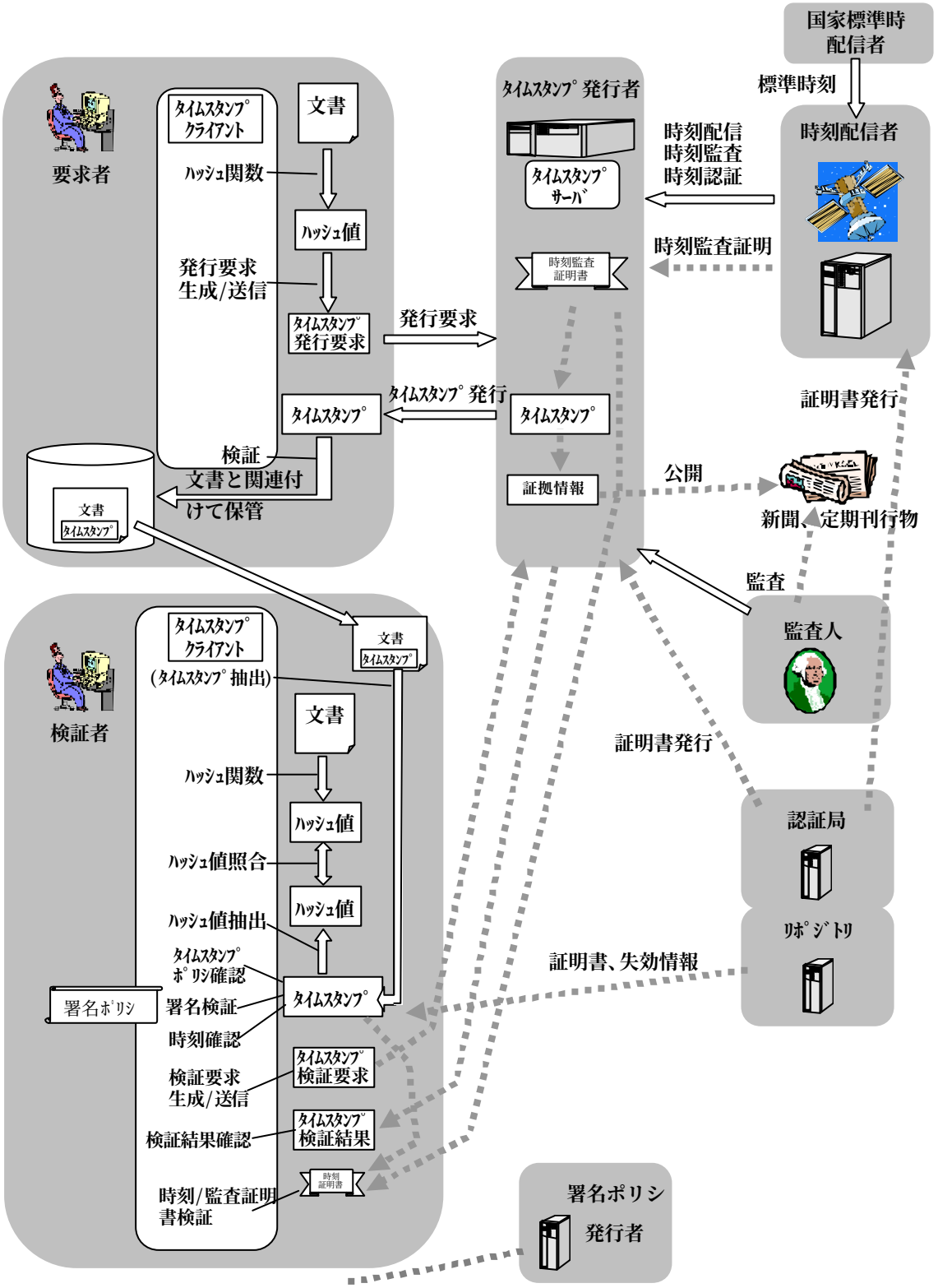


図 2-1 タイムスタンプの利用モデル

図 2-1 に基づき、タイムスタンプの取得、検証、及び時刻配信／監査／認証の手順を示す。

(1) タイムスタンプの取得

[要求者]

① 用途や目的に適ったタイムスタンプを発行できるタイムスタンプ発行者を選択する。このとき、対象が電子署名文書であり、タイムスタンプの要件が署名ポリシで示された場合、その要件を考慮する。

② タイムスタンプクライアントを用いてタイムスタンプ取得対象文書のハッシュ値を生成する。

③ タイムスタンプクライアントを用いてタイムスタンプ発行要求を作成し、タイムスタンプ発行者に送付する。タイムスタンプ発行者が複数のタイムスタンプポリシを持つ場合、タイムスタンプポリシの ID を指定するなどして、適当なタイムスタンプを要求する。

[タイムスタンプ発行者]

④ ハッシュ値と時刻からタイムスタンプを生成する。

⑤ 要求者にタイムスタンプを発行する。

[要求者]

⑥ タイムスタンプを受信し、検証した後に文書と関連付けて保管する。

(2) タイムスタンプの検証（手順はこのとおりとは限らない）

(a) 検証者自身で検証を行う場合

[検証者]

① タイムスタンプ、タイムスタンプ取得対象文書、電子署名文書に添付されたタイムスタンプを検証する場合は必要に応じて署名ポリシを取得する。

② タイムスタンプクライアントを用いてタイムスタンプを検証する。署名ポリシが存在する場合はそれとの整合性を踏まえて検証を行う。

②-1 必要に応じてタイムスタンプを文書から抽出する。

②-2 タイムスタンプのデジタル署名を検証する。デジタル署名が無効であればタイムスタンプも無効であると判断する。

②-3 タイムスタンプから時刻情報を抽出し、確認する。

②-4 タイムスタンプからタイムスタンプ取得対象文書のハッシュ値を抽出する。

②-5 タイムスタンプ取得対象文書からハッシュ値を生成する。

②-6 タイムスタンプから抽出したハッシュ値とタイムスタンプ取得対象文書からハッシュ値を照合する。照合に失敗した場合、タイムスタンプは無効であると判断する。

③ 必要に応じてタイムスタンプポリシの ID やポリシの内容を確認する。ID やタイムスタンプ発行者によりポリシが判断できるような場合は、逐一内容を確認する必要はない。

④ タイムスタンプクライアントによりタイムスタンプデータ内あるいはタイムスタンプ発行者から得た時刻監査証明書を検証する（時刻監査証明書が存在する場合）。検証に失敗した場合、タイムスタンプは無効であると判断する。

(b) 発行者に検証を依頼する場合

① タイムスタンプ、タイムスタンプ取得対象文書、電子署名文書に添付されたタイムスタンプを検証する場合は必要に応じて署名ポリシを取得する。

② タイムスタンプクライアントを用いてタイムスタンプ発行者にタイムスタンプ検証の代行を依頼する。

②-1 タイムスタンプ検証要求を作成し、タイムスタンプ発行者にタイムスタンプ検証要求を送付する。

[タイムスタンプ発行者]

②-2 タイムスタンプを検証し、検証結果を検証者に送付する。

[検証者]

②-3 タイムスタンプ検証結果を確認する。署名ポリシが存在する場合は、それとの整合性も確認する。

③ 必要に応じてタイムスタンプポリシの ID やポリシの内容を確認する。ID やタイムスタンプ発行者によりポリシが判断できるような場合は、逐一内容を確認する必要はない。

④ タイムスタンプクライアントによりタイムスタンプデータ内あるいはタイムスタンプ発行者から得た時刻監査証明書を検証する（時刻監査証明書が存在する場合）。検証に失敗した場合、タイムスタンプは無効であると判断する。

(3) 時刻配信／監査／認証

(a) タイムスタンプ発行者が時刻配信のみ受ける場合

[時刻配信者]

① タイムスタンプサーバを特定することなく、標準時刻を配信（ブロードキャスト）する。

[タイムスタンプ発行者]

② 専用の S/W や H/W により、配信された時刻を受信し、タイムスタンプサーバのクロックを調整する。

(b) タイムスタンプ発行者が時刻監査を受ける場合

[時刻配信者]

① 専用の S/W を用い、タイムスタンプサーバを特定し、タイムスタンプサーバのクロックの標準時刻とのズレを計測する。

② 監査結果や時刻監査証明書をタイムスタンプ発行者に送付する。

[タイムスタンプ発行者]

③ 監査結果を受け、必要な場合は(a)と同様な方法でタイムスタンプサーバのクロックを調整する。監査結果や時刻監査証明書を公開する場合もある。

(c) 時刻認証を受ける場合

[時刻配信者]

① 専用の S/W や H/W を用い、タイムスタンプサーバを特定した上でタイムスタンプサーバのク

ロックの標準時刻とのズレを計測する。必要に応じ、タイムスタンプサーバに時刻を配信し、タイムスタンプサーバのクロックを時刻配信者側からの指示により、調整して同期を取る。ズレが一定以上大きくなった場合、タイムスタンプサーバの動作を強制終了させる場合もある。

② 認証結果や時刻監査証明書をタイムスタンプ発行者に送付する。

[タイムスタンプ発行者]

③ タイムスタンプサーバの動作が強制終了された場合は、所定の対処を行う。認証結果や時刻監査証明書を公開する場合もある。

2.2 タイムスタンプの利用パターン

タイムスタンプは、データの存在時刻（データがある時刻以前に存在していたこと）とそれ以降、内容に変更がないことを証明するために用いられる。認証・公証 WG では、電子証明や電子署名文書の長期保存におけるタイムスタンプの重要性を主張してきた。そこで本節では、タイムスタンプの利用パターンとして、内容を示すデータに適用する場合、署名に適用する場合、署名文書の長期保存に適用する場合を区別し、それぞれの場合のタイムスタンプの意義を説明する。

①文書等のデータの存在時刻の証明

技術メモ、仕様書、図面、契約書、ログ、音声、マルチメディアなどのあらゆるデータの存在時刻を証明するためにタイムスタンプを用いることができる。文書等のデータに対してタイムスタンプを取得し、そのデータとタイムスタンプを関連付けて保管しておき、必要なときに検証を行う。タイムスタンプの検証により、それらのデータがタイムスタンプの示す時刻以前に存在し、それ以降、内容に変更がないことを確認できる。

②署名の存在時刻の証明

署名の存在時刻を証明するためにタイムスタンプを用いることができる。文書等のデータではなく、文書等のデータより生成した署名データに対してタイムスタンプを取得する。署名者が署名生成時に署名生成直後にタイムスタンプを取得する場合、署名検証者が署名検証時にタイムスタンプを取得する場合などが考えられる。いずれの場合も署名生成時刻からタイムスタンプ取得時刻までの間が短い（有効期限切れや失効が発生する以前）ことが望ましい。取得したタイムスタンプは署名データ内に格納するなど、両者を関連付けて保管する。タイムスタンプの検証により、有効期限切れが発生した場合、署名が有効期限切れ以前に存在したことを確認でき、失効が発生した場合、失効情報が得られる限り、署名が失効以前に存在したことを確認できる。

③署名文書の長期保存

署名文書の署名を長期間再検証可能な状態で保存するためにタイムスタンプを用いることができる。この場合、署名データ及び検証用の各種データに対して複数のタイムスタンプを取得し、それらを関連付けて保管する。署名データ及び検証用の各種データに加えてタイムスタンプを検

証することにより、元の署名と関連付けられた証明書の有効期限切れや失効の後であっても、当初署名が存在していた時点で、その署名が有効であったことを確認できる。詳細については、４章を参照のこと。

2.3 タイムスタンプシステムの構成例

タイムスタンプ発行者を中心とした、時刻配信者、要求者、及び検証者からなるタイムスタンプシステムの構成例を示す。

通常は、タイムスタンプ発行者及び時刻配信者には高い信頼性を要求されるため、図 2-2 に示すように両者が TTP（Trusted Third Party）であるような構成をとる。

現在タイムスタンプ発行者が利用できるタイムスタンプサーバシステムとして、RFC3161 に基づくタイムスタンプを発行するものであり、秘密鍵を外部に取り出すことが不可能かつ内部クロックを時刻配信者のみが調整できる（タイムスタンプ発行者自身は調整できない）ような耐タンパなハードウェア（H/W）装置が存在する。このような耐タンパ H/W 装置を用いる場合、時刻配信者側の装置との間で安全な相互認証を行った上で時刻配信者がタイムスタンプ装置のクロックの同期を取りかつそのことを保証することができるため、時刻配信者のみが TTP であれば良く、タイムスタンプ発行者が TTP である必要はなくなる。この場合、図 2-3 に示すようにタイムスタンプ発行者をタイムスタンプの要求者と同一の組織やドメインに置くような構成も可能となる。
勿論、耐タンパタイムスタンプサーバ H/W を TTP としてのタイムスタンプ発行者が利用しても良い（図 2-2）。

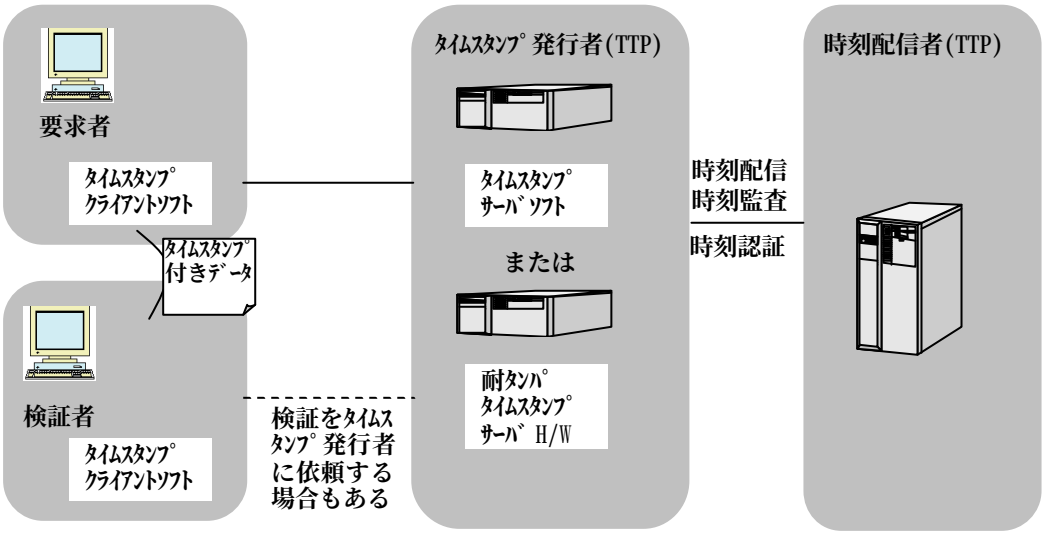


図 2-2 タイムスタンプ発行者及び時刻配信者が TTP である場合

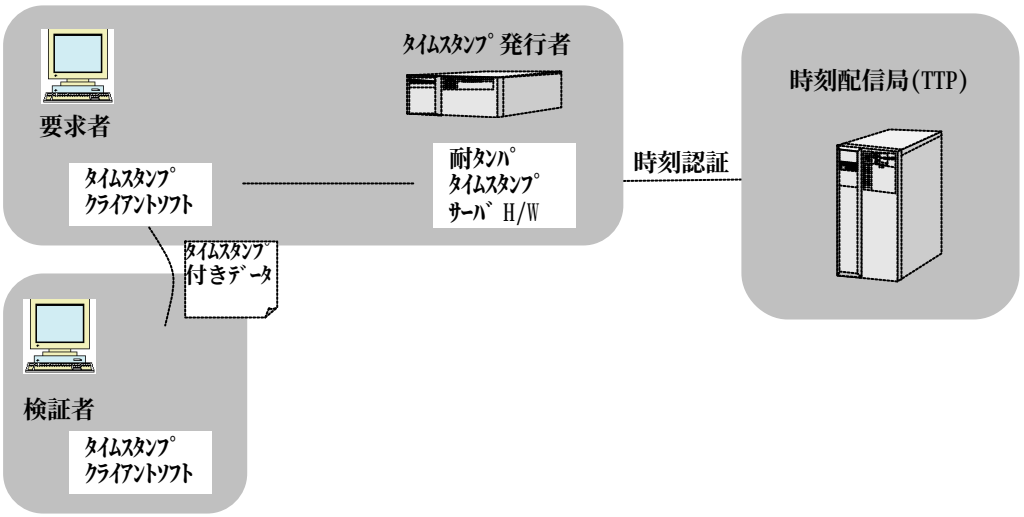


図 2-3 時刻配信者のみが TTP である場合

図 2-2 や図 2-3 では要求者や検証者などの利用者にタイムスタンプ発行者が直接対面するような構成を示したが、図 2-4 のように、文書管理や電子申請などのアプリケーションサービスの背後にタイムスタンプ発行者が位置するような構成も考えられる。この場合、タイムスタンプは、利用者の意識やコントロールの範囲外でアプリケーションサービスによって取得されることがある。また、アプリケーションサービスのクライアントソフトがタイムスタンプ検証機能を持ち、アプリケーションサービスが生成したタイムスタンプ付きデータのタイムスタンプの検証を行えるような場合も考えられる。

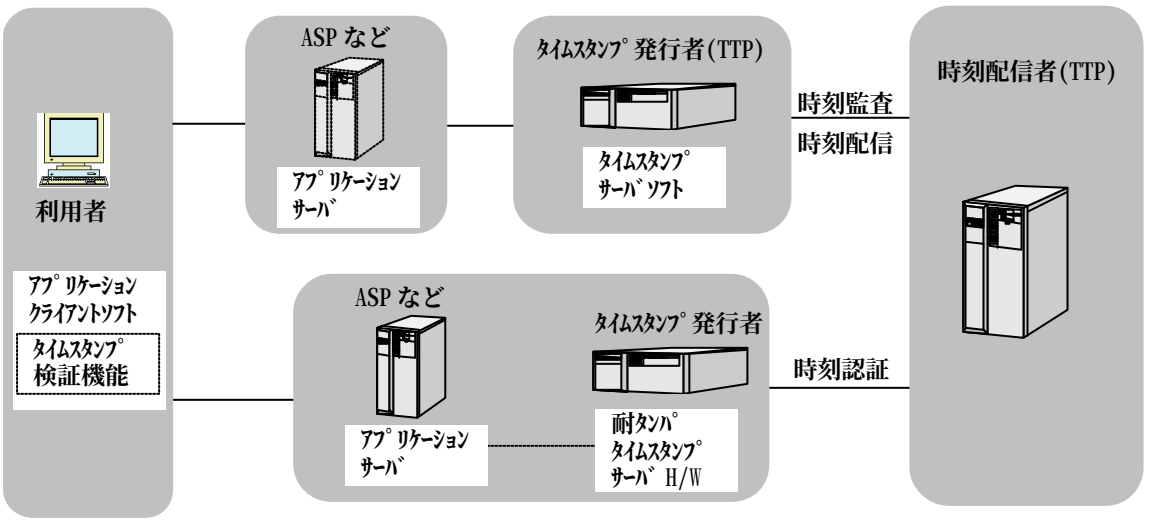


図 2-4 アプリケーションサービスがタイムスタンプを利用する場合

2.4 タイムスタンプを利用する製品およびサービス例

タイムスタンプサービスあるいはタイムスタンプ装置や、それらに対して時刻を配信／監査／同期を行うサービスのほか、タイムスタンプを利用する製品やサービスが既に市場に現れ始めている。その中からいくつかの例を次に挙げる。これらの例の一部の詳細説明が調査報告書に記載されているので、そちらも合わせて参照いただきたい。

・PDF 文書にタイムスタンプ付与する Acrobat プラグイン
・タイムスタンプ付与可能な電子カルテシステム
・タイムスタンプを付与することによる音声記録公証システム
・タイムスタンプを用いた建設業界向け電子契約サービス
・タイムスタンプを用いたデジタルコンテンツの著作権管理サービス
・タイムスタンプを利用して原本性を保証する文書保管サービス
・タイムスタンプによりデータの存在を証明する電子公証サービス

2.5 タイムスタンプを利用する標準例

タイムスタンプを利用する標準が、デジタル署名との関連でいくつか提案されている。これらは、「電子署名文書長期保存に関する中間報告（H13.3）」及び「電子署名文書長期保存に関するガイドライン（H14.3)」でも述べたデジタル署名持つ問題点を補完することを狙ったものである。なお、タイムスタンプそのものの標準については本書第５章あるいは調査報告書を参照いただきたい。

①長期署名フォーマット
・RFC3126, “Electronic Signature Formats for long term electronic signatures”
・Timestamp over signature、Archive timestamp の２通りのタイムスタンプを用い、デジタル署名の再検証を可能とする。
・RFC3161 のタイムスタンプを利用する。

②XML 版長期署名フォーマット
・ETSI TS 101 903, “XML Advanced Electronic Signatures(XAdES)”
・長期署名フォーマットの XML 署名版。
・RFC3161 のタイムスタンプあるいは５章に示す XML タイムスタンプを利用する。

③署名ポリシ
・RFC3125, “Electronic Signature Policies”
・署名者と検証者がデジタル署名を有効であるとみなすための規則を集めたもの。
・デジタル署名に添付されるタイムスタンプを有効とみなすための条件を含む。
・ETSI TR 102 038, “XML format for signature policies”では、署名ポリシの XML フォー

マットが規定されている。

④医用デジタル画像と通信におけるデジタル署名

・NEMA Standards Publication PS 3 Supplement 41 “DICOM Digital Signatures”

・DICOM データに対する署名フォーマットの規格であり、署名データ（PKCS#1 など）生成時に取得した署名データに対するタイムスタンプ（RFC3161）を DICOM 署名フォーマット内に格納できるように規定されている。

3．各種タイムスタンプサービスの特徴、比較表

3.1　比較対象

タイムスタンプには、「シンプル・プロトコル」、「リンキング・プロトコル」と「分散プロトコル」などの種類がある。ここでは製品化やサービス化が行われている「シンプル・プロトコル」と「リンキング・プロトコル」においてサービスについての特徴を比較する。

シンプル・プロトコルとリンキング・プロトコルの２つの方式は、利用者が電子文書のハッシュ値を送信し、TSA がタイムスタンプを返すという基本部分は同様の処理を行う。

次に２つのタイムスタンプの概要を述べる。

（1）シンプル・プロトコルを利用したサービス

このタイムスタンプは電子文書のハッシュ値に対して認証機関(CA)から公開鍵証明書の発行を受けた TSA が正確な時刻とともにデジタル署名したものである。

（2）リンキング・プロトコルを利用したサービス

このタイムスタンプには対象となっている電子文書のハッシュ値、時刻情報、タイムスタンプの正当性を証明するための必要情報（リンク情報など）が含まれている。特徴として全てのタイムスタンプはこれまで生成されたタイムスタンプに依存するように生成され、その過程で発生したリンク情報（ハッシュ値）を定期的に新聞等に公表することで、システムの信頼性を確保している。

3.2　比較表

下記の比較表は、タイムスタンプの利用者がタイムスタンプサービス選択時に必要とする情報を仮定した比較項目である。

	シンプル・プロトコルを利用したサービス（ＰＫＩベース）	リンキング・プロトコルを利用したサービス
有効期間	・　タイムスタンプ証明書の有効期間と同じ期間 ※例：RSA1024 の場合、５年程度	・　ハッシュ関数の強度に依存 ※タイムスタンプの有効期間を長くするために異なるハッシュ関数を用いる場合がある

タイムスタンプの検証	・ タイムスタンプのデジタル署名の検証に必要な証明書と CRL を取得できれば、TSA 以外の第三者による検証が可能である ・ タイムスタンプのデジタル署名の検証に必要な証明書と CRL がそろっていれば、オフラインによるローカル検証が可能である	TSA もしくは検証機関に検証を要求する必要がある
タイムスタンプに含まれる情報	・ 文書のハッシュ値 ・ 時刻 ・ TSA ポリシ ・ 順序性(ordering) ・ 時刻の精度(accuracy) など	・ 文書のハッシュ値、 ・ 時刻 ・ タイムスタンプの正当性を証明するための必要情報（リンク情報など）
信頼の拠所	・ハッシュ関数・公開鍵暗号に安全性を依存 ・認証機関を信頼の拠所とする	・ ハッシュ関数に安全性を依存 ・ リンク情報の公開・検証により信頼性を確保
信頼性低下、および、信頼喪失の可能性	・ TSA の証明書失効や秘密鍵の危殆化 ・ タイムスタンプの署名有効期間を超えて保証する場合、タイムスタンプ更新の失敗 ・ CA の秘密鍵の危殆化 ・ TSA の時刻の改ざん	・ システム内のハッシュ値の改ざん ・ リンク情報の消失 ・ TSA の時刻の改ざん
不正行為の可能性	・ TSA と署名者が結託することで、タイムスタンプの改ざんが可能である（耐タンパモジュールが必須） ※ 時刻証明を行う TA の属性証明書を利用することにより、TSA の時刻の改ざんを防止可能	・ TSA と利用者が結託することで、TSA に登録したという事実を抹消できる（ただし TSA のリンク情報を検証することで不正の事実を特定できる）
利用形態	・ローカル検証が可能であるため、大量にタイムスタンプの発行される（大量の検証処理が発生する）環境に適している	・比較的長期間保管が必要な電子文書の利用に適している

標準化	・ RFC3161(IETF PKIX WG) ・ ISO/IEC 18014-2：2002	・ ISO/IEC 18014-3：2002
システム構成	・クライアントと TSA 以外に、認証機関と連携したシステムの構成	クライアントと TSA(TSA には過去の履歴を保管するデータベースが必要)
コスト	・ タイムスタンプの利用 ・ タイムスタンプの更新とその回数	・ タイムスタンプの利用 ・ 検証処理を TSA や検証機関が行うため、タイムスタンプ検証時に費用が発生する可能性がある

※ 本比較表は、各方式の一般的な特徴を比較したものであり、アルゴリズムの改良や機能追加したものを対象としていない。

4．長期保存

タイムスタンプの利用場面のひとつとして、デジタル署名付き電子文書の長期保存での利用がある。本章では、デジタル署名付き電子文書の長期保存において、３章で述べた、「シンプル・プロトコル」に基づくタイムスタンプを利用する場合と、「リンキング・プロトコル」に基づくタイムスタンプを利用する場合の２つについて述べる。

4.1 シンプル・プロトコルの利用例

4.1.1 ETSI 長期署名フォーマット

シンプル・プロトコルのタイムスタンプを利用したデジタル署名付き電子文書の長期保存の実現にあたって、ETSI が規定するフォーマット（長期署名フォーマット）が用いられるケースがある。

長期署名フォーマットとは、公開かぎ証明書の有効期限切れ・失効、または、 暗号アルゴリズムの脆弱化後も、長期に渡ってデジタル署名の有効性を検証可能とするため、RFC2630（Cryptographic Message Syntax）で規定されている署名フォーマット（CMS SignedData）を拡張したものであり、ETSI TS 101 733 Electronic Signature Formats において規定されている。そして、これと同様な規定が、IETF においても RFC3126（Electronic Signature Formats for long term electronic Signatures）として規定されている。

以下に、長期署名フォーマット内部の各形式について示す。

(1) ES（the Electric Signature）

デジタル署名つき電子文書の基本フォーマットであり、署名者によって提供される署名ポリシ ID（Signature PolicyID)、その他の署名属性（Other Signed Attribute)、デジタル署名（Digital Signature）の３つからなる。

署名ポリシとは、署名者と検証者がデジタル署名を有効とみなすための、署名の生成と有効性検証の一連の規則であり、ETSI TS 101 733 Electronic Signature Formats において規定され、これと同様な規定が RFC3125（Electronic Signature Policies）として規定されている。

(2) ES-T（the ES with Timestamp）

ES に対して、TSA から取得した、ES の署名値（CMS の SignerInfo の SignatureValue）に対するタイムスタンプを追加したものであり、タイムスタンプの形式としてはシンプル・プロトコルの RFC3161 が使用される。このタイムスタンプは、署名者が署名の存在時刻を確定する場合や、検証者が署名の検証時刻を確定する場合に使用される。タイムスタンプを付与することによって、デジタル署名（ES）の存在について事後否認されることを防止できるようになる。

(3) ES-C（the ES with Complete validation data）

ES-T に対して、デジタル署名の検証の際に用いる認証パス上にある全ての証明書（ただし署名

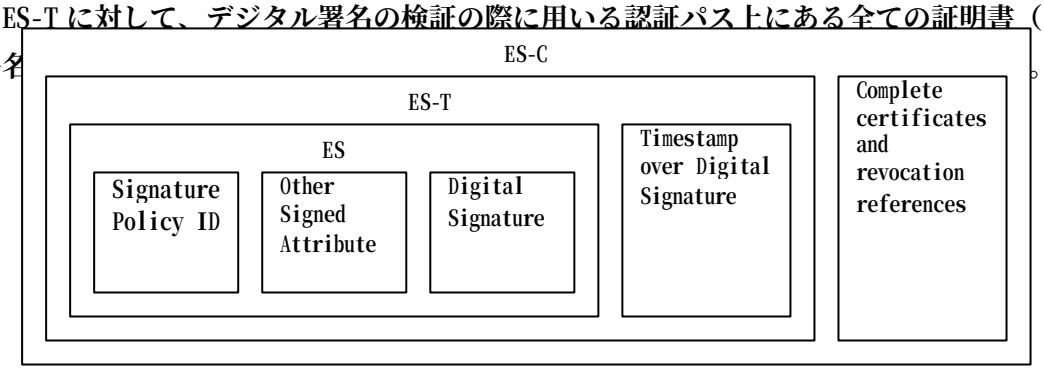


図 4-1 ES、ES-T、ES-C

(4) ES-X Long（the ES with Extended validation data）

ES-C に対して、認証パス上にある全ての証明書と CRL または OCSP 応答の値を追加したものである。署名の検証者がこれらのデータにアクセスできないことを想定して使用される。

なお、ES-X には、ES-X Long の他に ES-X Type1 と ES-X Type2 と呼ばれる署名フォーマットがあるが、H13 年度 ECom 認証公証 WG の長期署名保存 TF の報告書では、ES-X Long をデジタル署名付き文書の長

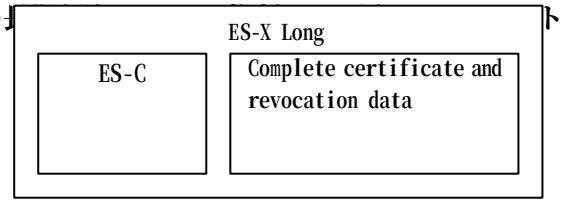


図 4-2 ES-X Long

(5) ES-A（the ES with Archived validation data）

ES-X Long に対して、デジタル署名とデジタル署名に追加した情報に対するタイムスタンプを追加したものである。このフォーマットはタイムスタンプの生成に必要な TSA の証明書の有効期限までは有効であるが、長期署名フォーマットの寿命を延長させるためには有効期限切れまでに T

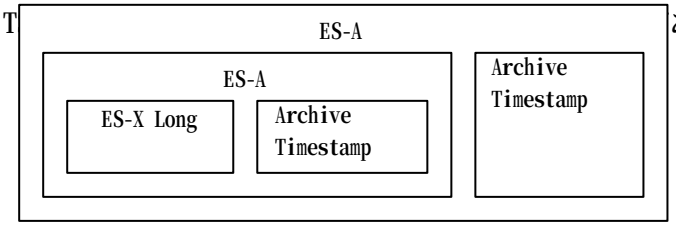


図 4-3 ES-A

4.1.2 処理の流れ

長期署名フォーマットに基づく、シンプル・プロトコルを利用したデジタル署名付き電子文書の長期保存処理の一例を以下の図に示す。

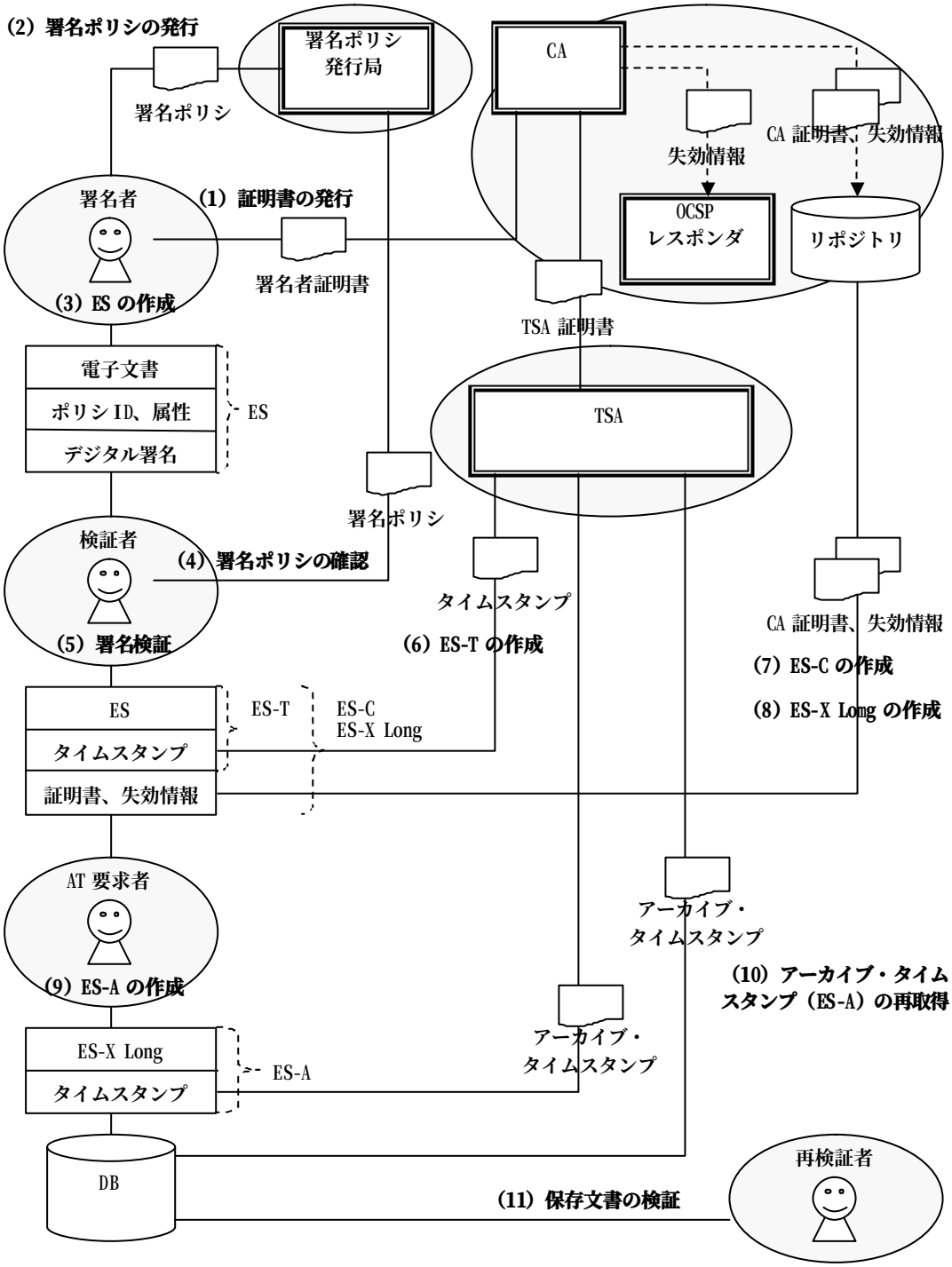


図 4-4 長期保存処理の流れ（シンプル・プロトコル）

構成エンティティは次の通りである。

・ 署名者

電子文書に対してデジタル署名を付与し、検証者へ提示する。

・ 検証者

デジタル署名付き電子文書の有効性を検証する。

・ 再検証者

デジタル署名付き電子文書の有効性を後日、検証する。

・ 署名ポリシ発行局

署名生成や署名検証を行う上での規則である署名ポリシを発行する。

・ CA／リポジトリ／OCSP レスポンダ

証明書、失効情報（CRL／OCSP 応答）を発行する。

・ TSA（タイムスタンプ発行局）

署名者、検証者から送信されたハッシュデータに対するタイムスタンプ（RFC3161 形式）を発行する。

・ AT（アーカイブ・タイムスタンプ）要求者

信頼される人格であり、アーカイブ・タイムスタンプ（ES-A）を TSA から繰り返し取得することで、デジタル署名付き電子文書の安全性を長期に渡って保証する。

4.1.3 処理内容

(1) 証明書の発行

署名者は CA に署名者証明書の発行を依頼する。

(2) 署名ポリシの発行

署名者は署名ポリシ発行局に要件に合致する署名ポリシの発行を依頼する。

(3) ES の作成

署名者は、署名ポリシ ID、および、その他の署名属性を含むデジタル署名を生成し、ES を作成する。そして、署名者は ES に署名者証明書を添付して、検証者へと転送する。

(4) 署名ポリシの確認

検証者は署名ポリシ発行局に対して､署名ポリシ ID に対応する署名ポリシの発行を依頼する。そして、取得した署名ポリシから、デジタル署名やタイムスタンプの有効性を検証する際に前提となる規則について確認する。

(5) 署名検証

検証者は、署名者証明書に対応する、認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を、リポジトリや OSCP レスポンダから取得し、署名者証明書の有効性を検証する。そして、署名者証明書を用いて、ES のデジタル署名の有効性を検証する。

(6) ES-T の作成

検証者は、検証時刻を確定するため、ES のデジタル署名のハッシュ値を TSA に転送して、タイムスタンプを取得する。同時に、タイムスタンプに含まれる TSA 証明書（ES-T）に対応する、認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を、リポジトリや OSCP レスポンダから取得して、TSA 証明書（ES-T）の有効性を検証する。そして、TSA 証明書（ES-T）を用いて、タイムスタンプの有効性を検証した後、ES にタイムスタンプを追加し、ES-T を作成する。

(7) ES-C の作成

検証者は、署名者証明書と TSA 証明書（ES-T）のそれぞれに対応する、認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）の参照値を ES-T に追加し、ES-C を作成する。

(8) ES-X Long の作成

検証者は、署名者証明書と TSA 証明書（ES-T）のそれぞれに対応する、認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を ES-C に追加し、ES-X Long を作成する。そして、これを AT 要求者の配下で、安全に保管する。

(9) ES-A の作成

AT 要求者は、ES-X Long 作成までに使用したアルゴリズムの脆弱化、または、TSA 証明書（ES-T）の有効期限切れが発生する前に、デジタル署名付き電子文書（ES）と、デジタル署名に追加した情報（ES-T、ES-C、ES-X Long）のハッシュ値を TSA に転送して、アーカイブ・タイムスタンプを取得する。同時に、アーカイブ・タイムスタンプに含まれる TSA 証明書（ES-A）に対応する、認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を、リポジトリや OSCP レスポンダから取得して、TSA 証明書（ES-A）の有効性を検証する。そして、TSA 証明書（ES-A）を用いて、タイムスタンプの有効性を検証した後、ES-X Long にアーカイブ・タイムスタンプを追加して、ES-A を作成し、これを安全に保管する。

(10)アーカイブ・タイムスタンプ（ES-A）の再取得

AT 要求者は、TSA 証明書（ES-A）が有効期限切れとなる前に、新たなアーカイブ・タイムスタンプ（ES-A）を取得する。これは、以下のような手順で行なう。

・ 署名ポリシの確認

署名ポリシ発行局に対して、署名ポリシ ID に対応する署名ポリシの発行を依頼する。そして、取得した署名ポリシから、デジタル署名の有効性を検証する際の前提となる規則について確認する。

・ 署名検証

ES-X Long 内に含まれる認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を用いて、署名者証明書の有効性を検証する。そして、署名者証明書を用いて、ES のデジタル署名の有効性を検証する。

・　アーカイブ・タイムスタンプ（ES-A）の検証

ES-A 内のアーカイブ・タイムスタンプに含まれる TSA 証明書（ES-A）に対応する、認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を、リポジトリや OSCP レスポンダから取得し、TSA 証明書（ES-A）の有効性を検証する。そして、TSA 証明書（ES-A）を用いて、ES-A 内のアーカイブ・タイムスタンプの有効性を検証する。

・　新たなアーカイブ・タイムスタンプ（ES-A）の取得

デジタル署名付き電子文書（ES）と、デジタル署名に追加した情報（ES-T、ES-C、ES-X Long、現世代以前のすべての ES-A）のハッシュ値を TSA に転送して、新たなアーカイブ・タイムスタンプを取得する。同時に、アーカイブ・タイムスタンプ内に含まれる TSA 証明書（ES-A）に対応する、認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を、リポジトリや OSCP レスポンダから取得し、TSA 証明書（ES-A）の有効性を検証する。そして、TSA 証明書（ES-A）を用いて、アーカイブ・タイムスタンプの有効性を検証した後、ES-A 内にアーカイブ・タイムスタンプを追加し、これを再度、安全に保管する。

(11)保存文書の検証

再検証者は、長期保存されたデジタル署名付き電子文書の検証が必要となった場合、以下のような手順で検証処理を行なう。

・　署名ポリシの確認

署名ポリシ発行局に対して、署名ポリシ ID に対応する署名ポリシの発行を依頼する。そして、取得した署名ポリシから、デジタル署名やタイムスタンプの有効性を検証する際の前提となる規則について確認する。

・　署名検証

ES-X Long 内に含まれる認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を用いて、署名者証明書の有効性を検証する。そして、署名者証明書を用いて、ES のデジタル署名の有効性を検証する。

・　タイムスタンプ（ES-T）の検証

ES-T 内のタイムスタンプに含まれる TSA 証明書（ES-T）に対応する、ES-X Long に含まれる認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を用いて、TSA 証明書（ES-T）の有効性を検証する。そして、TSA 証明書（ES-T）を用いて、ES-T 内のタイムスタンプの有効性を検証する。

・　アーカイブ・タイムスタンプ（ES-A）の検証

最新世代の ES-A 内のアーカイブ・タイムスタンプに含まれる TSA 証明書（ES-A）に対応する、認証パス上の CA 証明書および失効情報（CRL／OCSP 応答）を、リポジトリや OSCP レスポンダから取得し、TSA 証明書（ES-A）の有効性を検証する。そして、TSA 証明書（ES-A）を用いて、最新世代の ES-A 内のアーカイブ・タイムスタンプの有効性を検証する。

4.1.4 TSA の選定における考慮事項

長期署名フォーマットの ES-T および ES-A 内に含まれるタイムスタンプを発行する TSA は、少なくとも以下のような点について考慮した上で選定することが望ましい。

(1) 信頼できる運用ポリシの確認

３章の比較表で挙げられているように、シンプル・プロトコルに基づくタイムスタンプを発行する TSA には、TSA 証明書の失効や秘密鍵の危殆化、そして TSA の時刻改ざんなどを要因とする信頼性の低下・喪失が発生する可能性や、TSA と署名者が結託することによるタイムスタンプの改ざんといった不正行為が発生する可能性がある。

そのため、TSA の利用者は TSA を選定するに当たって、上記のような問題が発生しないように、TSA が信頼できる運用ポリシを設けていることを確認する必要がある。また、運用ポリシの妥当性を確認するにあたっては、ETSI TS 102 023：“Policy requirements for Time-Stamping Authorities（タイムスタンプ局のポリシ要件）”といった標準化された仕様書を参考にするとよい。

(2) 署名アルゴリズムと鍵長

タイムスタンプに付与される TSA の署名のアルゴリズムとしては、RSA、DSA、ECDSA などが挙げられる。現在、PKI を利用した多くの商用サービスにおいて、RSA は 1024bit 以上、DSA は 1024bit、ECDSA は 160bit（RSA で 1024bit 相当）以上の鍵長で利用されている。そのため、タイムスタンプに付与される TSA の署名も、上記と同様な鍵長を使用したものであることが望ましい。

なお、上記アルゴリズムの中でも、標準的に使用されている RSA の場合、1024bit と 2048bit の鍵長では、署名処理時間にして約８倍の時間差があることが確認されている。そこで、タイムスタンプに関わる処理時間を少しでも短くしたい場合には 1024bit の署名鍵を使用した TSA を採用し、またタイムスタンプの安全性を確保したい場合にはさらに長い鍵長の署名鍵を使用した TSA を採用するというように使い分けるとよい。

(3) ハッシュアルゴリズム

タイムスタンプトークン内に含まれるハッシュの生成や、TSA の署名の生成に用いられるハッシュアルゴリズムとして、SHA-1 と MD5 が挙げられる。しかし、近年、MD5 は署名法等において安全性に関する問題が指摘されているので、現時点ではハッシュアルゴリズムには MD5 ではなく SHA-1 を利用することを推奨する。

4.2 リンキング・プロトコルの利用例

4.2.1 SecurePod サービス

リンキング・プロトコルを利用したデジタル署名付き電子文書の長期保存のモデルが NTT データから SecurePod サービスとして実装・提案されている。

SecurePod サービスでは、PKI の電子認証の技術と、TSA である SecureSeal のリンキング・プロトコルに基づくタイムスタンプの技術によって、保存エージェントである SecurePod サーバがデジタル署名付き電子文書の長期保存を実現する。

具体的には、SecurePod サーバが、原本であるデジタル署名付き電子文書、公開かぎ証明書、CRL、Root 証明書の原本性と、公開かぎ証明書の有効性を検証することで、電子文書に付与したデジタル署名が署名時に有効であったかを証明するというものである。

4.2.2 処理の流れ

リンキング・プロトコルを利用したデジタル署名付き電子文書の長期保存処理の流れを以下の図に示す。

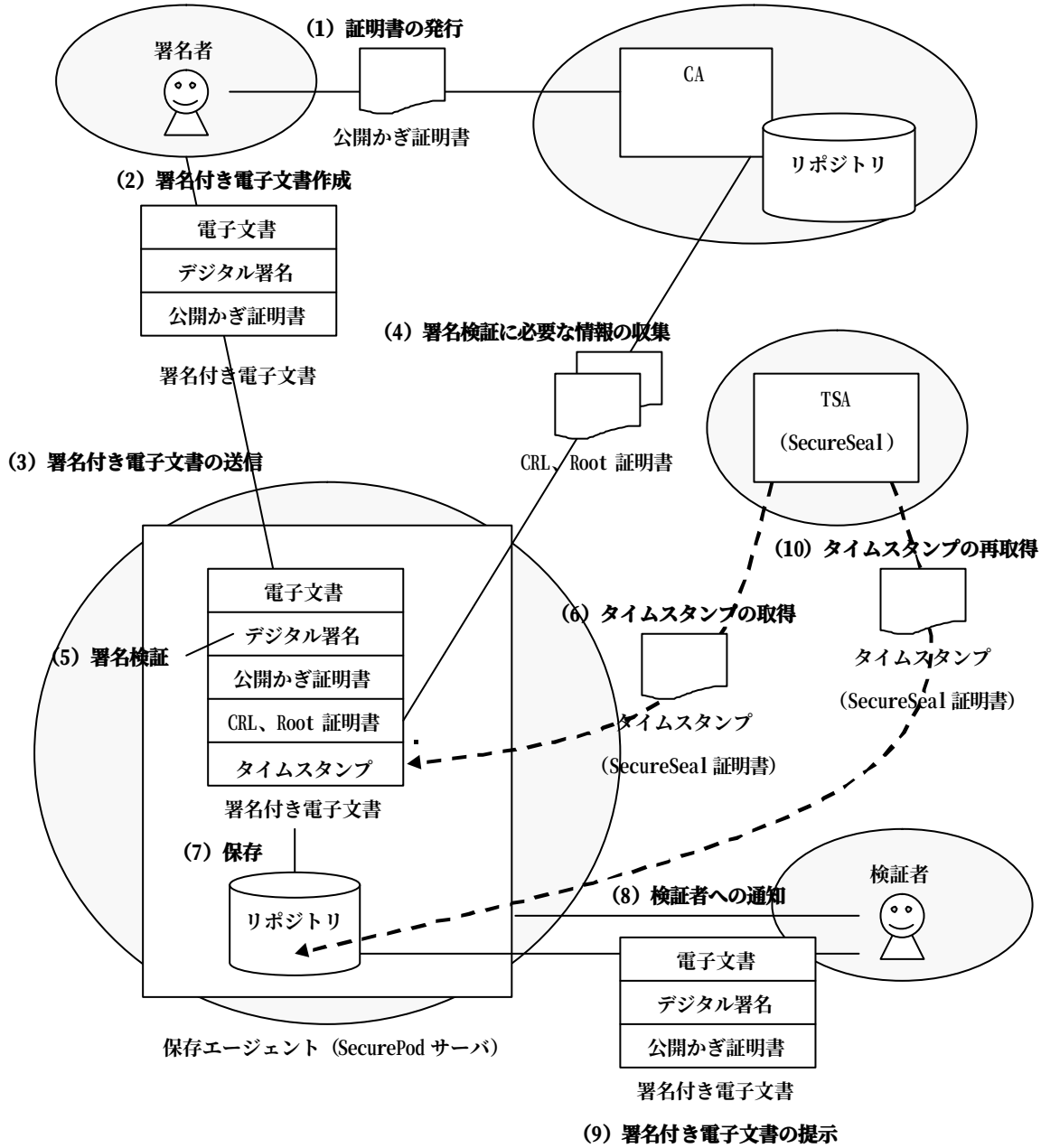


図 4-5 長期保存処理の流れ（リンキング・プロトコル）

構成エンティティは次の通りである。

・　署名者

デジタル署名付き電子文書を作成し、これを署名者の公開かぎ証明書と共に保存エージェントに送信する。

・　検証者

保存エージェントからデジタル署名付き電子文書を入手する。

・　保存エージェント（SecurePod サーバ）

署名者から入手したデジタル署名付き電子文書の有効性検証と長期保存を行なう。

・　CA/リポジトリ

公開かぎ証明書、CRL を発行する。

・　TSA（SecureSeal）

リンキング・プロトコルに基づいたタイムスタンプ（SecureSeal 証明書）を発行する。

4.2.3 処理内容

(1) 証明書の発行

署名者は CA に署名者の公開かぎ証明書の発行を依頼する。

(2) 署名付き電子文書の作成

署名者は公開かぎ証明書に基づき、電子文書に対するデジタル署名を作成する。

(3) 保存エージェントへの署名付き電子文書の送信

署名者は保存エージェントに対して、デジタル署名付き電子文書と公開かぎ証明書を検証者の宛先と共に送信する。

(4) 署名検証に必要な情報の収集

保存エージェントはデジタル署名の有効性を検証するために必要な情報を収集する。必要な情報は、公開かぎ証明書に対応する CRL と、CRL に付与された CA の署名を検証するための Root 証明書である。

(5) 署名検証

保存エージェントは（4）で入手した CRL、Root 証明書を用いて、(3) で入手した公開かぎ証明書の正当性、有効性を検証した後、公開かぎ証明書を用いて、署名付き電子文書に付与されたデジタル署名の有効性を検証する。

(6) タイムスタンプの取得

保存エージェントはデジタル署名付き電子文書と公開かぎ証明書を結合したもののハッシュ、CRL のハッシュ、および、Root 証明書のハッシュを作成し、これらを TSA に送信し、それぞれのタイムスタンプを取得する。

(7) 保存

保存エージェントはデジタル署名付き電子文書、公開かぎ証明書、CRL、Root 証明書、および、タイムスタンプを自身のリポジトリ内に安全に保存する。

(8) 検証者への通知

保存エージェントは検証者に対して保存エージェント上に検証者宛のデジタル署名付き電子文書が保存されていることを通知する。

(9) 署名付き電子文書の提示

保存エージェントは保存されたデジタル署名付き電子文書について､以下の検証を行ない、有効性を確認した後、検証者に提示する。

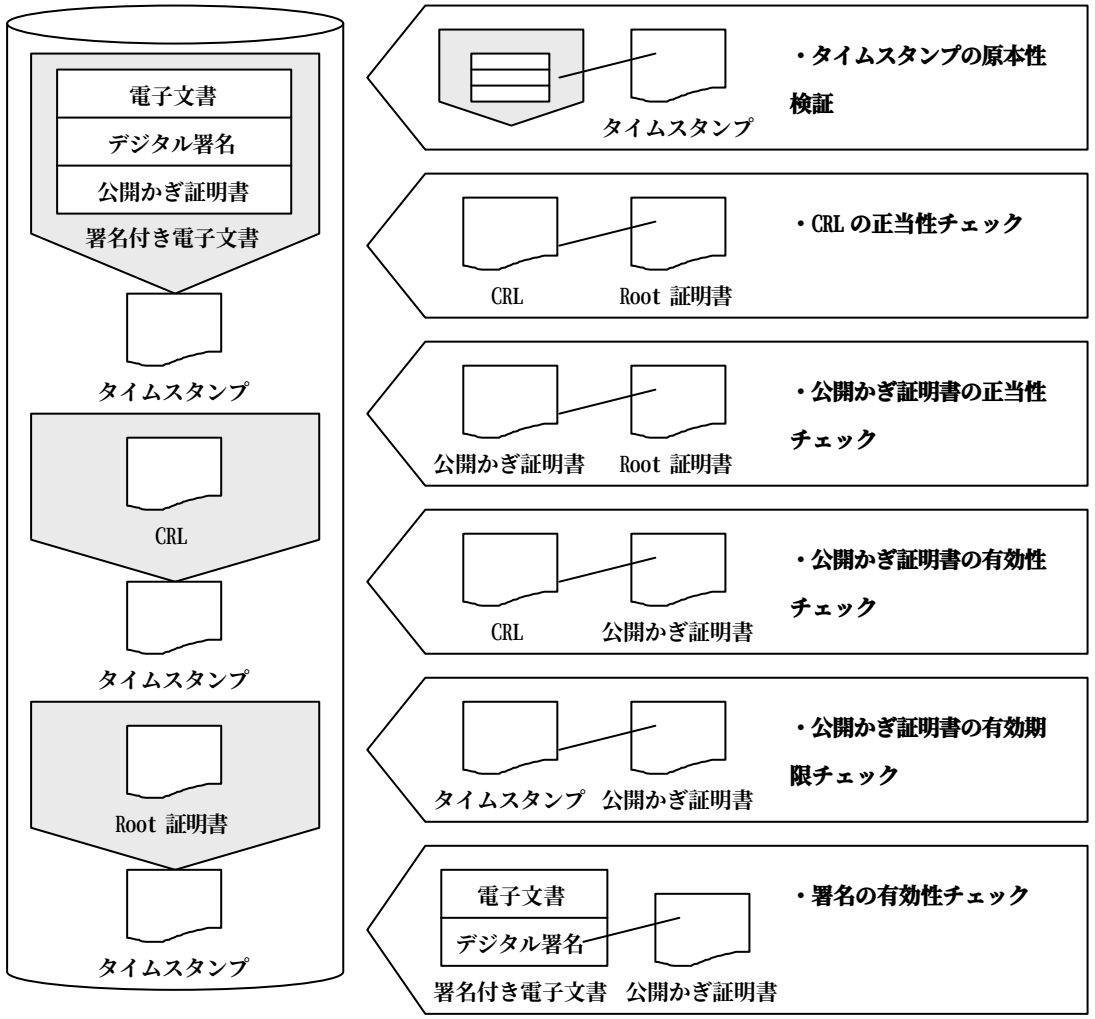


図 4-6 保存エージェントによる保存文書の有効性検証

・　タイムスタンプ（SecureSeal 証明書）の原本性検証

タイムスタンプを用いて、デジタル署名付き電子文書、公開かぎ証明書、CRL、Root 証明書の原本性を検証する。これにより、デジタル署名付き電子文書、公開かぎ証明書、CRL、Root 証明書が登録時点から改ざんのないことが証明できる。

・　CRL の正当性チェック

CRL に付与された CA の署名を Root 証明書で検証する。これにより、CRL が CA から発行されたものであることを証明できる。

・　公開かぎ証明書の正当性チェック

公開かぎ証明書を Root 証明書で検証する。これにより、公開かぎ証明書の所有者（署名者）の身元を証明できる。

・　公開かぎ証明書の有効性チェック

公開かぎ証明書のシリアル No が CRL にないことを確認する。これにより、署名時点で公開かぎ証明書が失効されていなかったことを証明できる。

・　公開かぎ証明書の有効期限チェック

タイムスタンプと公開かぎ証明書の有効期限を比較する。これにより、署名者が公開かぎ証明書の有効期限内に署名を行なったかどうか証明できる。

・　署名の有効性チェック

電子文書に付与されたデジタル署名を公開かぎ証明書で検証する。これにより、公開かぎ証明書の所有者（署名者）が署名を作成したことが証明できる。

(10)タイムスタンプの再取得

タイムスタンプを生成する段階で保存エージェントや TSA が使用している SHA-1 や MD5 などのハッシュアルゴリズムは、時間の経過と共に危殆化される可能性が考えられる。その結果、保存エージェント内に保管されているすべてのデータとタイムスタンプの信頼性は時間の経過と共に低下してしまう。

このようにハッシュアルゴリズムが危殆化されそうになった場合には、保存エージェントおよび TSA は速やかにハッシュアルゴリズムの更新を行ない、そして、保存エージェントは過去に保管されたすべてのデータとタイムスタンプを結合したものに対するハッシュを生成し、これを TSA に送信し、新たなタイムスタンプの取得を行なう。

これにより、保存エージェントはハッシュアルゴリズム危殆化後も過去に保管されたすべてのデータの原本性を保証することが可能となる。

5．デジタル署名とタイムスタンプ利用上の留意点

デジタル署名に対するタイムスタンプの利用上の留意点を考察する。タイムスタンプを署名に付与するときの理解のために、現在標準として定められている署名フォーマットを概観し、署名にタイムスタンプをどのように付けるのか、長期署名フォーマットの標準について述べ、またタイムスタンプの利用上の留意点をふまえた利用ガイドについても述べることにする。さらに今後XML 文書などについて、XML タイムスタンプを付けることが検討されているが、この XML タイムスタンプの動向についても述べる。また長期署名保存の観点から XML タイムスタンプ付き XML 長期署名フォーマットについての動向を述べることにする。

5.1 署名フォーマット

タイムスタンプと署名との関係を説明するためにまず、署名フォーマットについて簡単に見ておくことにする。署名フォーマットには以下に述べるようにプリミティブ署名、CMS SignedData、Long term signature format(ASN.1)、XML 署名、Long term signature format(XML)など複数の形式がある。用途に応じて使い分ける必要がある。タイムスタンプを署名に付ける場合それぞれについて個別に対応することになる。

5.1.1 プリミティブ署名

X.509 証明書や CRL への署名に使われている署名フォーマットで、図 5-1 に示すように被署名データと署名アルゴリズムおよび署名値からなる最もシンプルな形式のものである。以下のASN.1 で定義されている。このような署名形式は署名アルゴリズム以外の一切の属性を持たないことから X509 の署名や PKI プロトコルの署名など特定用途の署名に用いられるもので、ビジネス目的などの永続的な署名には用いられない。文書などの永続的な署名には後に述べる CMS 署名やXML 署名が用いられる。

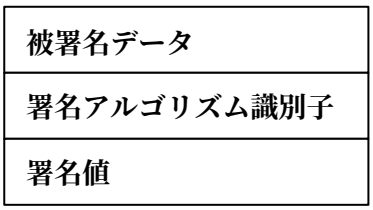


図 5-1 プリミティブ署名

```
SIGNED { ToBeSigned } ::= SEQUENCE {
   toBeSigned  ToBeSigned,                 --被署名データ
   algorithm   AlgorithmIdentifier,        --署名アルゴリズム識別子
   signature   BIT STRING                  --署名値
}
AlgorithmIdentifier  ::=  SEQUENCE  {
   algorithm               OBJECT IDENTIFIER,
   parameters              ANY DEFINED BY algorithm OPTIONAL  }
```

この OBJECT IDENTIFIER で指定される署名アルゴリズムは RSA、DSA、ECDSA などがあるが、RSA 方式の場合は sha1WithRSAEncription や md5WithRSAEncription などで指定される。署名値は被署名データ（ToBeSigned ）のハッシュ値とパディングを PKCS#1（注）のフォーマットにエンコードしたものに署名鍵で暗号化した値である。RSA の場合、AlgorithmIdentifier の parameters は NULL とする。（RSA の場合は Public Key 自身に Modulus n と parameter である Public exponent e を含んでいるからである）。DSA や ECDSA ではパラメータは陽に指定する。

（注）PKCS#1 の RSA 署名フォーマットについて

RSA 署名の基本フォーマットは PKCS#1 で定められているが、これにはアドホックなパディングを用いている PKCS#1 v1.5 と確率論的に破ることが難しいということが証明できる方式 RSA-PSS（Probabilistic Signature Scheme）署名フォーマット PKCS#1 v2.1 がある。PKCS#1 v1.5 のフォーマットはアドホックなパディングを用いているが致命的な欠陥はいままでに指摘されていない。PKCS#1 v1.5 は RSA 署名フォーマットとして多く使われており、各種の標準がこれをサポートしている。今後は RSA-PSS 署名フォーマットが用いられるようになろうが、まだ多くの実装がないことから現在の段階では互換性に問題がある。

日本の電子署名法の省令では特定認証業務で使用するハッシュ関数は SHA-1 のみとし、MD5 は今後認めないことになった（１年間の猶予期間をおく）。また RSA-PSS を新たにオプションで採用することになった。

5.1.2 CMS SignedData

IETF SMIME WG で定められた署名フォーマットで、旧 RFC2630 が改定され、RFC 3369（フォーマット）、RFC3370（アルゴリズム）と２つの RFC で規定されている。これはフォーマット規定とアルゴリズム規定を分離することで、アルゴリズムの追加、改定に伴ってフォーマット規定を改定しなくてもよいようにするためである。

Cryptographic Message Syntax (RFC 3369)

Cryptographic Message Syntax (CMS) Algorithms (RFC 3370)

CMS は S/MIME の署名形式として標準化されたものであるが、一般の署名フォーマットとして使われる。SignedData タイプは PKCS#7 の SignedData タイプと基本的に同一である（署名値が PKCS#7 の場合は BitString に対して、CMS SigedData では Octed String の違いがある）。このフォーマットはプリミティブ署名だけではなく、署名対象（eConten）に加えて検証のために証明書を添付することや署名者情報(SignerInfo)として署名データに加えて署名する署名属性をつけられる。署名にタイムスタンプを付ける場合は非署名属性のフィールドを用いる。

CMS SigedData の署名フォーマットの概念図を図 5-2 に示す。

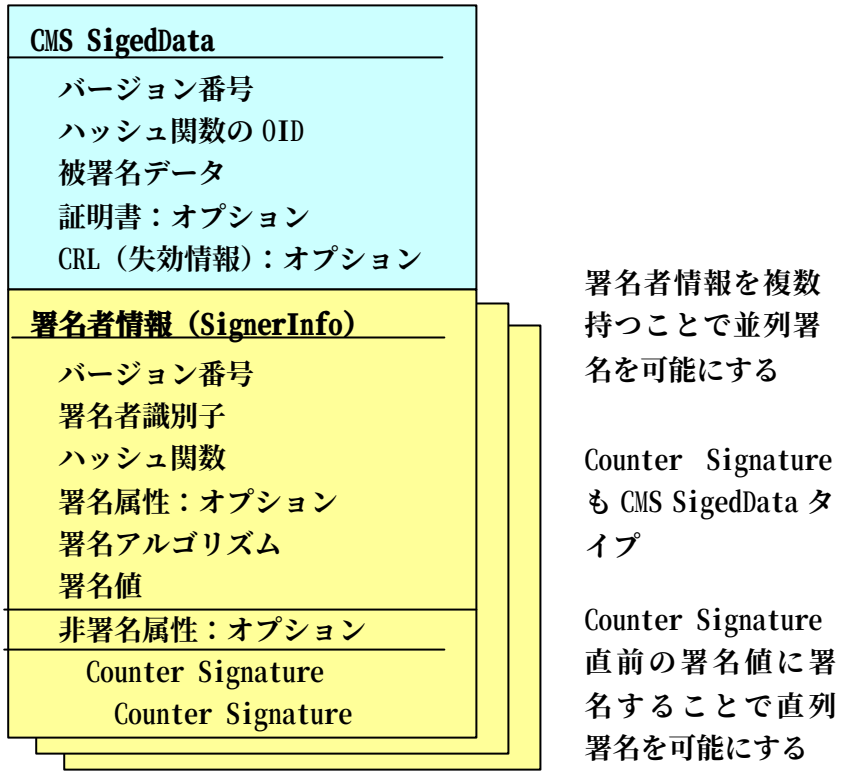


図 5-2 CMS SigedData 構造

以下に CMS SigedData の ASN.1 形式の概要を示す。

```
ContentInfo ::= SEQUENCE {
  contentType SignedData,                          --OID
  content [0] content of SignedData Type }

SignedData ::= SEQUENCE {                          --署名データタイプ
  version CMSVersion,
  digestAlgorithms DigestAlgorithmIdentifiers,   --ハッシュ関数の指定
  encapContentInfo EncapsulatedContentInfo,      --被署名データ
  certificates [0] IMPLICIT CertificateSet OPTIONAL,  --添付証明書
  crls [1] IMPLICIT CertificateRevocationLists OPTIONAL,
  signerInfos SignerInfos }                        --署名者情報

    DigestAlgorithmIdentifiers ::= SET OF DigestAlgorithmIdentifier
    SignerInfos ::= SET OF SignerInfo

EncapsulatedContentInfo ::= SEQUENCE {               --被署名データ
  eContentType ContentType, --Data Content Type, SignedData Content Type
  eContent [0] EXPLICIT OCTET STRING OPTIONAL }      --署名対象データ

    ContentType ::= OBJECT IDENTIFIER

SignerInfo ::= SEQUENCE {                            --署名者情報
  version CMSVersion,
  sid SignerIdentifier,                              --署名者識別子
```

```
digestAlgorithm DigestAlgorithmIdentifier,          --ハッシュ関数
signedAttrs [0] IMPLICIT SignedAttributes OPTIONAL, --署名属性
signatureAlgorithm SignatureAlgorithmIdentifier,    --署名アルゴリズム
signature SignatureValue,                           --署名値
unsignedAttrs [1] IMPLICIT UnsignedAttributes OPTIONAL }  --非署名属性
```

この形式の署名フォーマットは署名対象（EncapsulatedContentInfo）が１つしか指定できないので、EncapsulatedContentInfo の eContent で指定される OCTET STRING に対して丸ごと署名される。もし、eContent で指定したもの以外のデータを別途指定して署名データに加える場合は署名属性で指定する。署名属性としては署名者が主張する署名時間（セキュアタイムスタンプではない）や署名者の Role などがある。非署名属性には CounterSignature や TimeStamp Token などが対象となる。CertificateSet には最低限署名者の証明書を付けることが推奨される。

CMS SignedData の複数署名。

・並列署名は SignerInfo を複数指定して実現する。

・直列署名は非署名属性（UnsignedAttributes）に CounterSignature（SignerInfo タイプ）を次々に付けることで何重にも直列署名を可能にする。

CMS SignedData タイムスタンプ

・セキュアタイムスタンプトークンは非署名属性（UnsignedAttributes）に置く。

5.1.3 長期署名フォーマット（ASN..1）

EU の電子署名法に適合するものとして ETSI TS 101 733 として定められた長期署名の再検証が可能な署名フォーマットである。これはまた IETF で Informational RFC として署名フォーマットと署名ポリシの２つに分けて発行されている。

Electronic Signature Formats for long term electronic signatures (RFC 3126)

Electronic Signature Policies (RFC 3125)

この形式のフォーマットは長期署名の再検証を可能とするために、署名フォーマットの中にタイムスタンプや最初に署名検証した際に用いた証明書や失効情報への参照や認証パス上のすべての証明書や失効情報を証拠として添付しておくものである。さらに長期記録において署名に用いた署名鍵が危殆化した場合にも署名当時の署名の有効性を検証可能にするためにフォーマット全体に更にタイムスタンプを付けることも定めている。

署名フォーマットは用途に応じて基本署名フォーマット（ES）、タイムスタンプ付き署名フォーマット（ES-T）、完全署名フォーマット（ES-C）、拡張署名フォーマット（ES-X）、署名記録のフォーマット（ES-A）を定めている。以下の図 5-3 にこれらの包含関係を示す。

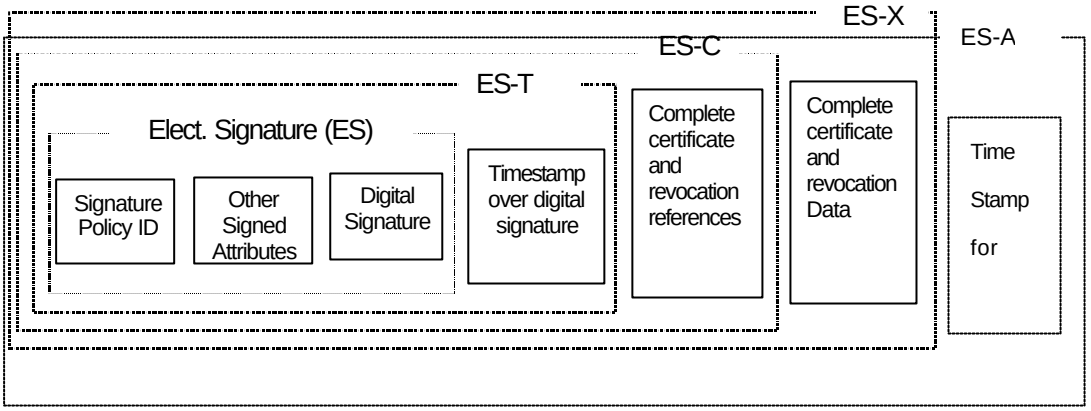


図 5-3 長期署名フォーマット

このフォーマットの特徴は長期署名フォーマットを定めた中で、ES に署名ポリシを必須で指定するとしていることである。署名ポリシは署名生成と署名検証に当って署名者と検証者が合意すべき事項を記載するとしている。署名ポリシは文章による表記であっても良いが、マシンリーダブルな構文として表現し署名がこのポリシに一致しているかどうかを自動的に処理することを可能にするポリシ構文も定義していることである。

このフォーマットについては平成 13 年度 ECOM 認証公証 WG の長期署名保存 TF の報告書に概要が述べられている。

長期署名フォーマット作成上の留意点

長期署名フォーマットを後に正しく検証するために認証パスの各証明書と失効情報については対象署名者の署名用証明書の認証パスだけではなく、タイムスタンプ認証パスやもし失効情報に OCSP レスポンスを用いる場合は OCSP レスポンダの認証パスも以下のように記録しなければならない。これは後の検証者がタイムスタンプや OCSP レスポンダの検証も正しく行えるようにするためである。

また、アーカイブタイムスタンプについても以下に述べる注意点が必要である。

1．ES-T のタイムスタンプは ES の署名値に対するタイムスタンプである。タイムスタンプサーバ証明書からタイムスタンプの信頼点までの認証パスについて、それぞれの証明書と失効情報（CRL または OCSP レスポンス）の参照値を ES-C に載せる。またタイムスタンプ認証パスの各証明書と各失効データを ES-X Long に載せておく。

2．失効情報として OCSP レスポンスを用いる場合、OCSP レスポンダ証明書から OCSP レスポンダの信頼点までの認証パス上の各証明書と各失効情報の参照値を ES-C に載せる。また認証パス上の各証明書と失効データは ES-X Long に載せておく。

3．これらのタイムスタンプや OCSP の認証パスの参照値や完全なデータは対象署名用証明書の認証パスと同様に、それぞれ CMS SignedData の Unsigned Attribute に置かれる。

4．ES-A のアーカイブタイムスタンプは内部の ES-T のタイムスタンプと違って、ES-X Long

のすべての各要素および以前に付けたアーカイブタイムスタンプに対するタイムスタンプである。ETSI のフォーマットはこのアーカイブタイムスタンプの認証パスの証明書や失効情報の参照値や完全なデータを記録しておく領域が定義されていないので記録しなくともよい（これについては 5.4.3 長期署名フォーマットの検証の項で再考する）。ただし、タイムスタンプサーバの証明書はタイムスタンプトークンに添付しておく。

これは最初のアーカイブタイムスタンプは ES-T が有効なうちに付けておく、ES-A のタイムスタンプが生きている（有効である）時はこのタイムスタンプの検証はオンラインで可能であるからである。次のアーカイブタイムスタンプを付ける時は、前のアーカイブタイムスタンプが有効であり、対象とする ES-A のフォーマットが有効であることを検証して行うのである。

5.1.4 XML 署名

XML 署名フォーマットの標準は W3C と IETF の共同の WG で策定された。

XML-Signature Syntax and Processing (RFC 3275)

XML 署名フォーマットは ASN.1 の署名フォーマット CMS SignedData を XML 構文に置き換えたものではなく XML の柔軟性を生かした独自のフォーマットを規定したものである。XML 署名フォーマットは CMS SigedData フォーマットと違い、署名者情報を複数持たず単一の署名者に対する署名フォーマットになっている。複数の並列署名を実現するためには以下に述べる Enveroped 署名の形式を用い、その中に複数の署名要素を収めることになる。XML 署名の基本構造は図 5-4 に示すようなものになる。

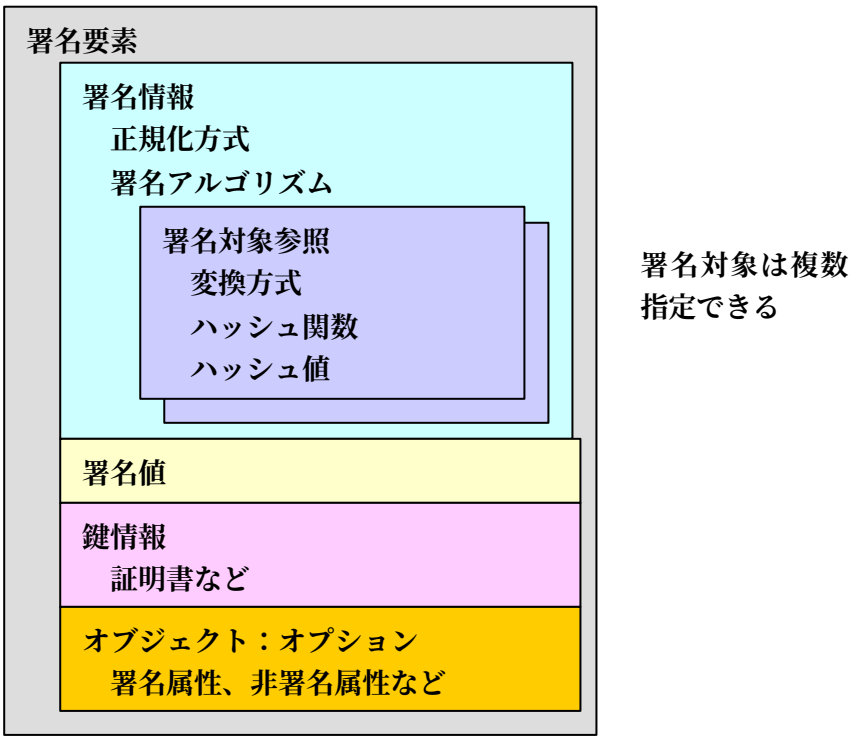


図 5-4 XML 署名構造

XML ディジタル署名は署名アルゴリズムや証明書やディジタル署名のタグを定め、指定した文書の範囲（複数可）に対して署名を付けることを可能にしている。CMS 署名フォーマットの場合は指定したデータコンテント全体に署名するのでコンテントの一部にコメントなどを加えると署名改竄になってしまうが、XML 署名は署名対象領域以外であれば任意のコメントなどを挿入できる特徴を持っている。

XML ディジタル署名を行うためには XML 構文と表現の同一性をはかるため、ディジタル署名をつける前に XML 構文の正規化処理（Canonicalization）を行う必要がある。この正規化処理も標準化されている（RFC 3076）。XML 署名は XML 文書だけでなく任意のオブジェクトに署名を付けることもできる。RFC3275 で示している XML 署名の構文は以下のようなものである。（ここでは？は 0 または 1、＋は 1 以上、*は 0 以上を示す）

```
<Signature ID?>                    --XML 署名
  <SignedInfo>                     --署名情報
    <CanonicalizationMethod/>      --正規化方式
    <SignatureMethod/>             --署名アルゴリズム
    (<Reference URI? >             --署名対象参照
      (<Transforms>)?              --変換方式
      <DigestMethod>               --ハッシュ関数
      <DigestValue>                --署名対象のハッシュ値
    </Reference>)+
  </SignedInfo>
  <SignatureValue>                 --署名値
 (<KeyInfo>)?                      --署名検証情報(証明書など：オプション)
 (<Object ID?>)*                   --署名属性（オプション）
</Signature>
```

XML 署名フォーマットは署名情報（SignedInfo）と署名値（SignatureValue）からなり、オプションとして署名検証情報（KeyInfo）と署名属性（Object）を付加えることができる。
署名検証情報（KeyInfo） には公開鍵や X.509 の証明書や PGP 証明書情報を指定することもできる。署名属性（Object）には署名に追加すべき付加情報や関連情報を指定できる。

XML 署名には図 5-5 に示すように３つの形式がある。

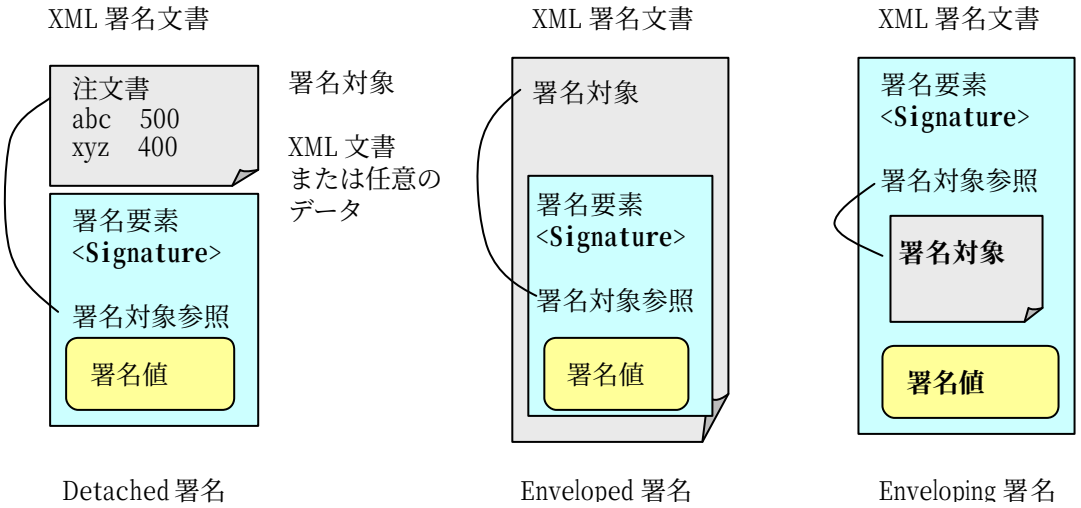


図 5-5 XML 署名の 3 つの形式

Detached 署名：署名要素<Signature>の外部にあるデータを署名参照<Reference>する形式で、XML 文書以外の任意のデータ（バイナリも可）を署名対象とする形式
Enveloped：署名対象要素の子要素に署名要素<Signature>を含む形式
Enveloping：署名要素<Signature>の子要素に署名対象要素を含む形式

XML 署名のタイムスタンプ

XML 署名にタイムスタンプを付ける標準的なスキーマはこの標準では規定していない。したがって独自なスキーマ規定を設けてタイムスタンプトークンを付ける必要があるが、以下に述べる ETSI の長期署名フォーマット(XAdES)で定義している XAdES-T のフォーマットが標準的なタイムスタンプ付与の形式となる。

XML 署名の複数署名

複数署名の観点から、XML 署名と CMS SignedData の署名フォーマットの違いは以下の点である。

・CMS SignedData には SignerInfo が複数指定できるので、並列署名が仕様に含まれている。これに対して XML 署名では<Signature>要素は単一の署名のフォーマットになっており、並列署名の場合は Detached 署名または Enveloped 署名の形式で同一の署名対象に対して複数の<Signature>要素を設ける必要がある。
・CMS SignedData には SignerInfo の Unsigned Attribute に Counter Signature を置いて直列署名を可能にしている。しかし XML 署名ではこのような標準の仕様が無い。従って、XML 署名で直列署名をサポートするためには<Object>要素で独自の Counter Signature を指定するようにするか，ETSI の XML 長期署名フォーマットが定義している<Object>要素の<Unsigned Properties>の<CounterSignature>を用いる必要がある。

並列署名

XML の並列署名は Detached 署名または Enveloped 署名形式で、同一署名対象を参照する複数の<Signature>要素を並べることで実現できる。この形式を以下の図 5-6 に示す。

同一署名対象に署名

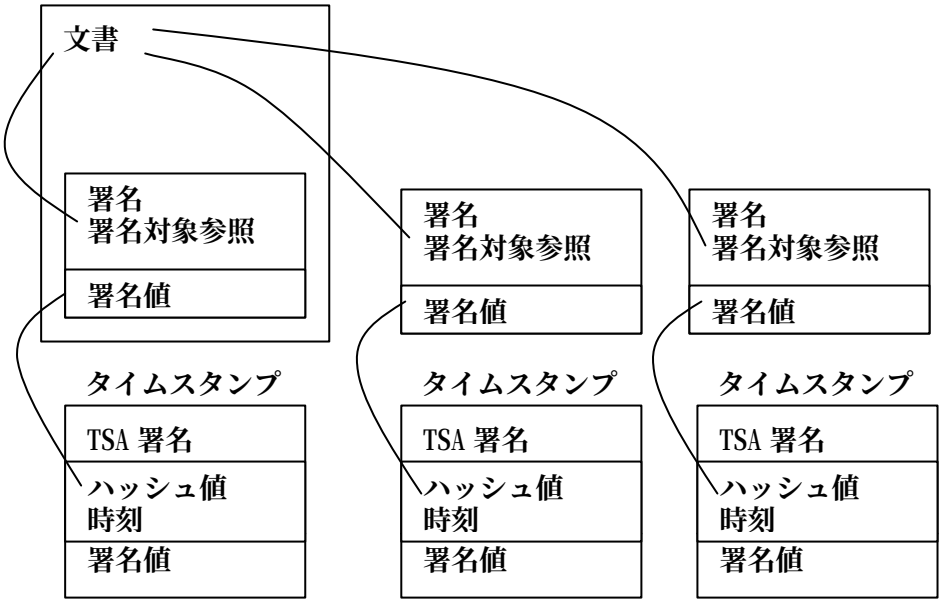


図 5-6 並列署名とタイムスタンプ

直列署名

直前の署名要素の署名値を署名対象にした署名要素を重ねることで順序付きの多重署名を実現できる。しかし、XML 署名標準 RFC3275 では直列署名を実現する Counter Signature のスキーマ定義がない。直列署名を可能にするために以下に述べる XML 長期署名フォーマットでは <Signature>要素で定義する<Object>要素の子要素に<UnsigedPropaties>に CMS と同様な <CounterSignature>要素を定義し、最初の署名の次の署名を<CounterSignature>におき、次からの署名を直前の<CounterSignature>の署名値に対する<CounterSignature>とする入れ子方式をとっている。直列署名とタイムスタンプの関係を図 5-7 に示す。

直前の署名値に署名

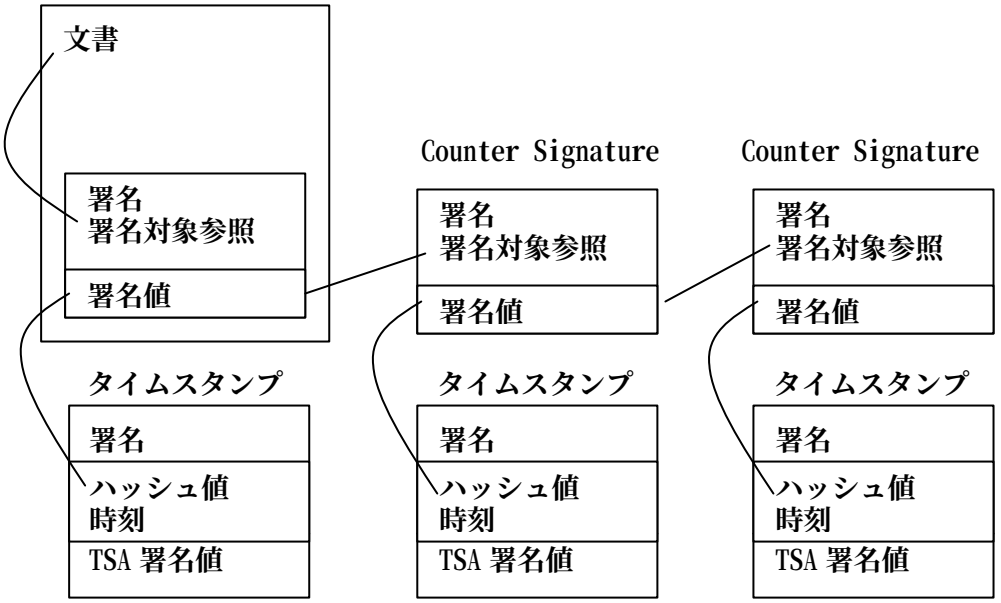


図 5-7 直列署名とタイムスタンプ

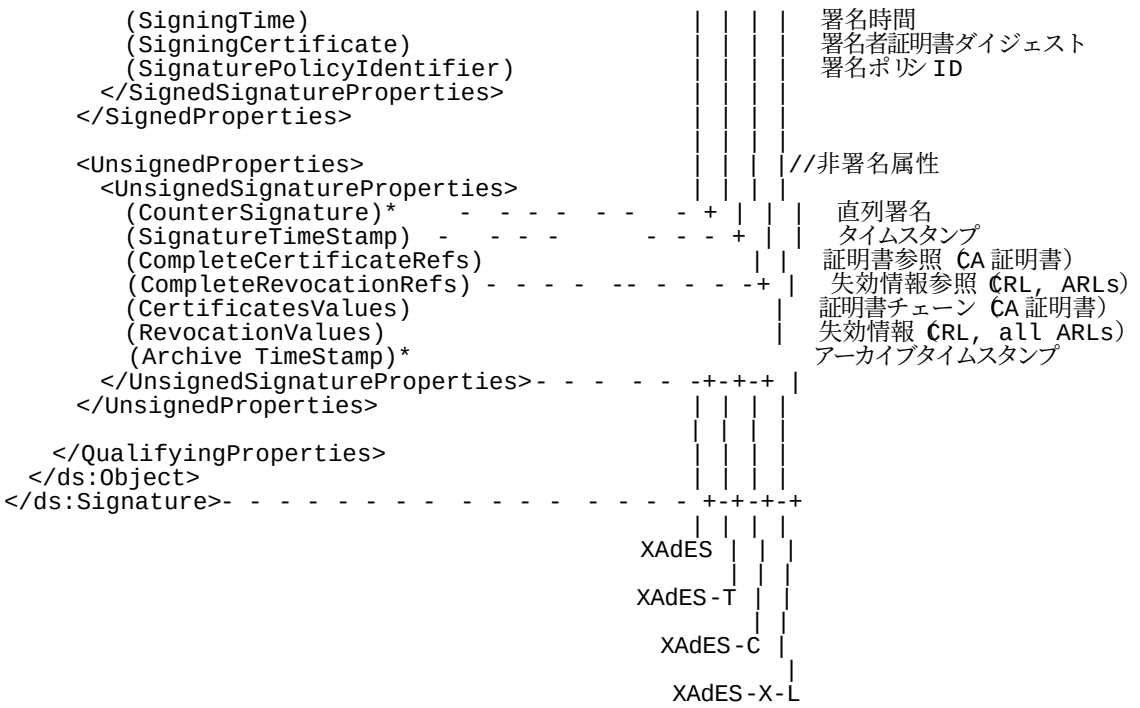
5.1.5 XML 長期署名フォーマット

ETSI では 2002 年 2 月に XML ベースの長期署名フォーマットを ETSI TS 101 903 XML Advanced Electronic Signatures (XAdES) として策定した。これは ASN.1 ベースの長期署名フォーマット（ES TS101 733）を XML ベースで定義したフォーマットである。ここでは ASN.1 ベースのタイムスタンプと（将来定める）XML タイムスタンプのどちらかを使用することのしている。
また XML 署名ポリシも XML format for signature policies - TR 102 038 (April 2002)として規定している。

以下に昨年 TF5 において ASN.1 ベースの長期署名フォーマットのガイドラインに沿った XML 長期署名フォーマットのプロファイルを示す。

Minimum XAdES-X-L format

```
                                XMLDISG
                                   |
<ds:Signature ID?>- - - - - - - - - - +- - - -+-+-+-+
  <ds:SignedInfo>                  |         | | | |//署名情報
    <ds:CanonicalizationMethod/>   |         | | | |
    <ds:SignatureMethod/>          |         | | | |
   (<ds:Reference URI? >           |         | | | |
     (<ds:Transforms>)?            |         | | | |
      <ds:DigestMethod>            |         | | | |
      <ds:DigestValue>             |         | | | |
    </ds:Reference>)+              |         | | | |
  </ds:SignedInfo>                 |         | | | |
  <ds:SignatureValue>              |         | | | |//署名値
  <ds:KeyInfo>   - - - - - - - -   -      +         | | | |//鍵情報 署名者証明書）
                                             | | | |
  <ds:Object>                                | | | |
    <QualifyingProperties>                   | | | |
                                             | | | |
      <SignedProperties>                     | | | |//署名属性
        <SignedSignatureProperties>          | | | |
```



? 0 or 1
* 0 or more
+ 1 or more

上記 XML ベースの長期署名フォーマットについても署名へのタイムスタンプ（XAdES-T）の認証パス（証明書チェーン）と、もし OCSP レスポンスが使われる場合には OCSP レスポンダ証明書の認証パスの扱いは ASN.1 の長期署名フォーマットで述べた留意点があてはまる。

5.2 タイムスタンププロトコル

シンプルプロトコルに基づくタイムスタンププロトコルは ASN.1 ベースの標準が IETF で策定されている（RFC3161）。XML 署名を含む XML 文書にタイムスタンプをつける場合、ASN.1 のタイムスタンプトークンをカプセル化（base64 でエンコード）して XML 文書に付与することもできるが、XML で定義したタイムスタンププロトコルの方がより柔軟に扱うことが出来る。最近（2002.10）、OASIS において DSS TC が形成されここで XML タイムスタンプ標準を検討することになった。ここでは ASN.1 ベースの標準 RFC3161 と XML タイムスタンプの概要について述べることにする。

5.2.1 RFC3161 ASN.1 タイムスタンプ

タイムスタンプには IETF で RFC3161 として ASN.1 で定義した標準プロトコル（TSP）がある。タイムスタンプはタイムスタンプ要求と、TSA のタイムスタンプ応答に添付されたタイムスタンプトークンからなる。

タイムスタンプ要求は、対象とするデータのハッシュ値を TSA に送ることで行われる。、TSA はタイムスタンプ応答に処理のステータスを示し、タイムスタンプトークンを返す。

タイムスタンプトークンは要求対象データのハッシュ値に TSA が信頼できる時刻源から得た

UTC タイムを加えて、さらに TSA ポリシや精度、順序性を加えて CMS SigenData 形式で TSA の署名を付けたものである。

署名データに対するタイムスタンプは対象データの署名値のハッシュ値に対してタイムスタンプを付けることにする。

タイムスタンプ要求

RFC3161 で定めているタイムスタンプ要求は以下の ASN.1 フォーマットに規定される。基本的に要求データのハッシュ値を提供することである。

```
TimeStampReq ::= SEQUENCE  {
   version              INTEGER  { v1(1) },
   messageImprint       MessageImprint,
     --a hash algorithm OID and the hash value of the data to be
     --time-stamped
   reqPolicy             TSAPolicyId              OPTIONAL,
   nonce                 INTEGER                  OPTIONAL,
   certReq               BOOLEAN                  DEFAULT FALSE,
   extensions            [0] IMPLICIT Extensions  OPTIONAL  }

   MessageImprint ::= SEQUENCE  {
        hashAlgorithm      AlgorithmIdentifier,
        hashedMessage      OCTET STRING  }

      TSAPolicyId ::= OBJECT IDENTIFIER
```

ここではタイムスタンプ要求のフォーマットとして以下のような要求フォーマットを指定することを推奨する（ETSI のタイムスタンプポリシー要求による）。

- ここで、reqPolicy はオプションであるが、指定しなければタイムスタンプ応答のトークンに TSA が指定するタイムスタンプポリシの OID が返される。
- nonce はリプレーアタックを防ぐためのもので、オプションであるが指定しておくべきである
- certReq はオプションでデフォルト False であるが True とすべきである
- extensions は指定しない

タイムスタンプ応答

タイムスタンプ応答は以下の図 5-8 に示すようにステータスとタイムスタンプトークンからなる。

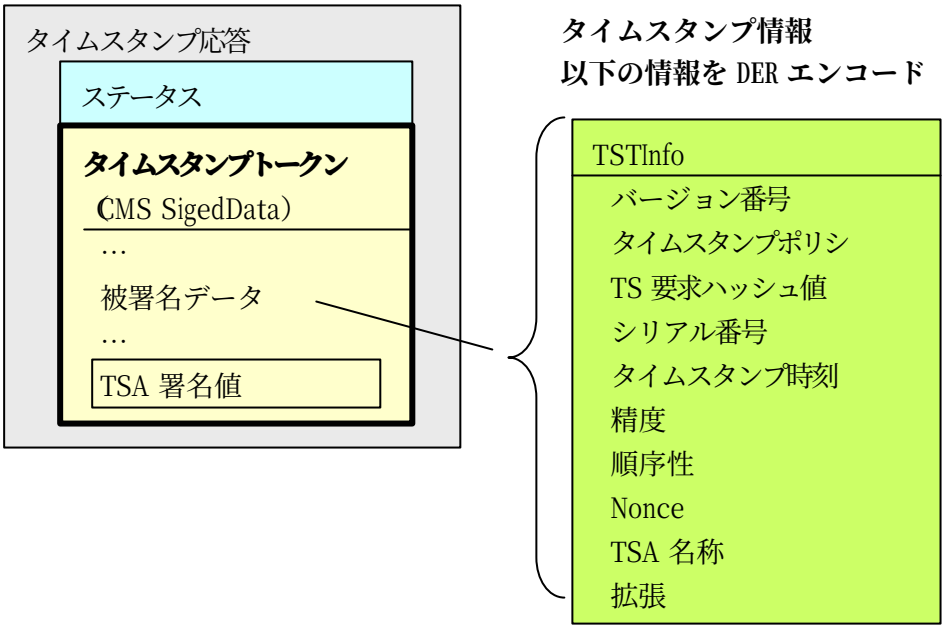


図 5-8 タイムスタンプ応答

以下に ASN.1 のタイムスタンプ応答の構文を示す。

```
TimeStampResp ::= SEQUENCE  {
   status                  PKIStatusInfo,
   timeStampToken          TimeStampToken     OPTIONAL  }
```

timeStampToken がオプションになっているのは status がエラーの場合は status のみが返され、timeStampToken は返されないからである。

Status のフォーマットは RFC2510 の PKIStatus の構文に従うこととする。

```
TimeStampToken ::= ContentInfo
  -- contentType is id-signedData ([CMS])
  -- content is SignedData ([CMS])
```

タイムスタンプトークンは CMS SignedData 構文で TSA のデジタル署名が付けられる。CMS SignedData で指定する EncapsulatedContentType は以下に示す TSTInfo を DER エンコードした値になる。

```
TSTInfo ::= SEQUENCE  {
   version                      INTEGER  { v1(1) },
   policy                       TSAPolicyId,
   messageImprint               MessageImprint,
     -- MUST have the same value as the similar field in
     -- TimeStampReq
   serialNumber                 INTEGER,
```

```
  -- Time-Stamping users MUST be ready to accommodate integers
  -- up to 160 bits.
genTime                 GeneralizedTime,
accuracy                Accuracy                OPTIONAL,
ordering                BOOLEAN             DEFAULT FALSE,
nonce                   INTEGER                 OPTIONAL,
  -- MUST be present if the similar field was present
  -- in TimeStampReq.  In that case it MUST have the same value.
tsa                     [0] GeneralName         OPTIONAL,
extensions              [1] IMPLICIT Extensions OPTIONAL  }
```

TSTInfo の構成は以下に示すプロファイルを推奨する（ETSI の TSA ポリシ要求から）。

・policy はこの TSA が持つポリシ OID を返す
・messageImprint は要求で指定されたものを返す
・genTime は UTC タイムを指定する
・accuracy はオプションであるがタイムスタンプポリシで定める値を返す
・ordering もオプションであるが DEFAULT FALSE のままでよい
・nonce は要求で指定されてものを返す
・tsa は TSA 名称を使うが、CMS SignedData の SinerInfo に書かれるので指定しなくて良い
・extentions は用いない

5.2.2 XML タイムスタンプ（TIML）

XML ベースのタイムスタンプは 2002 年 10 月から OASIS DSS TC で標準化の検討に入った。以下に示す TIML（Temporal Integrity Markup Language）は XML タイムスタンプの候補として Entrust が提案しているものである。TIML は RFC3161 を比較的忠実に XML スキーマに置き換えたものを想定しているが RFC3161 と同値ではない。TIML 以外にも XML ベースのタイムスタンプフォーマットの候補が提案されており、最終的な標準として以下に示す TIML も変更されるかもしれない。

タイムスタンプ要求

タイムスタンプ要求は以下に示す構文となる。

```
<TimeStampRequest>
  <ts:Policy ID=URI></tst:Policy>        //タイムスタンプポリシ、URI で指定
  <ts:Digest>                            //タイムスタンプ要求データのハッシュ値
    <ds:Transforms>...</ds:Transforms>
    <ds:DigestAlgolithm>
    <ds:DigestValue>
  </ts:Digest>
  <ts:Nonce>...</ts:Nonce>               //Nonce　オプション
  <ts:Extensions>...</ts:Extensions>     //拡張　オプション
</TimeStampRequest>
```

タイムスタンプ応答

タイムスタンプ応答は、タイムスタンプステータス<StatusInfo>とタイムスタンプ情報

<TimeStampInfo>を含む Enveloping Signature およびオプションとしての拡張からなる。

```
<TimeStampResponce>
  <ts:StatusInfo>...</ts:StatusInfo>       //応答のステータス
  <ds:Signature>                           .//タイムスタンプトークン：オプション
    …. //以下に示す TimeStampInfo を含む Enveloping Signature
  </ds:Signature>
  </ts:Extensions>....</ts:Extensions>     //拡張：オプション
</TimeStampResponce>
```

応答のステータスは正常、拒否などの状態情報とエラーの理由からなる。正常以外のステータスの場合はタイムスタンプトークンである<ds:Signature>は返されない。

タイムスタンプトークンはタイムスタンプ情報（TimeStamoInfo）を署名対象とした Envelopinng Signature の形式となっている。

タイムスタンプ情報（TimeStamoInfo）

```
<TimeStampInfo>
  <ts:Policy ID=URI></ts:Policy>                          //タイムスタンプポリシ
  <ts:Digest> … <ts:Digest>                        //要求されたハッシュ値
  <ts:SerialNumber>...</ts:SerialNumber>           //シリアル番号、オプション
  <ts:CreationTime> … </ts:CreationTime>           //タイムスタンプ時刻
  <ts:Accuracy> … </ts:Accuracy>                          //精度：オプション
  <ts:Ordering> … </ts:Ordering>                          //順序性：オプション
  <ts:Nonce> … </ts:Nonce>                         //Nonce：オプション
  <ts:Extensions> … </ts:Extensions>               //拡張：オプション
</TimeStampInfo>
```

タイムスタンプ時刻、精度、順序性は ASN.1 形式のタイムスタンプ RFC3161 と同じ意味である。RFC3161 にあるバージョン番号やTSA の名称がないのは、バージョンは ts 名前空間を指定する URI で示され、TSA 名称は署名<Signature>に含まれるので冗長性を除いた。

5.3 タイムスタンプの利用法

タイムスタンプは時刻に加えて対象データのハッシュ値に TSA 署名が付けられていることから、TSA の署名を検証することで、

- 対象データがタイムスタンプ時刻以前に存在していたこと（データの存在証明）
- 対象データのハッシュ値とタイムスタンプトークンに含まれるハッシュ値を比較することで対象データの完全性の検証機能を持つこと（データ公証）

を証明できる。タイムスタンプの付されたデータは以下のように署名無し文書へのタイムスタンプと署名データへのタイムスタンプがある。

5.3.1 署名無し文書へのタイムスタンプ

文書のハッシュ値に対するタイムスタンプによって、以下のことが検証できるようになる。

・文書がタイムスタンプ時刻以前に存在していたこと

・文書の改竄の検出

5.3.2 署名付き文書のタイムスタンプ

文書に署名が付けられたことによって、文書作成者の本人確認機能を有し、公証機能が強化される。すなわち、署名付き文書のタイムスタンプによって、以下のことを示すことが出来る。

・署名がタイムスタンプ時刻以前に存在していた

・署名の改竄の検出

・署名名義人の確認

5.3.3 署名文書のタイムスタンプは誰が付けるのか？

署名文書に対してタイムスタンプは署名者が付ける場合と検証者が付ける場合がある。署名者がタイムスタンプを付けた場合でも検証者が付けた場合でも 3.2 で述べたタイムスタンプの効果は同じである。タイムスタンプ付与者を誰にするかはポリシに依存する。ETSI の長期署名フォーマット仕様のタイムスタンプ付き署名（ES-T）ではどちらが付けても良いことになっており、署名者がタイムスタンプを付けなかった場合には最初の署名検証者がタイムスタンプを付けることとしている。

タイムスタンプ付き署名フォーマット ES-T では、署名用証明書の有効期限内であれば、署名検証者が失効情報（CRL または OCSP）の失効日時（revokedDate）とタイムスタンプ時刻を比較することで署名が失効以前に存在していたかどうか、すなわち署名がタイムスタンプ時点で有効であったかどうかを検証できる。

署名用証明書の有効期限が過ぎた場合は、失効情報（CRL 等）を取得する手段が無くなっているかもしれないので、署名文書を電子公証機関に預けるか、ETSI の長期署名フォーマット（ES-X long）を最初の署名検証者が作っておかなければならない。

5.3.4 署名文書へのタイムスタンプの付け方

署名文書に対するタイムスタンプには以下に示す 2 つのタイムスタンプの付け方がある。すなわち、

・「文書＋署名」全体のハッシュ値に対するタイムスタンプ

・「署名値」のハッシュ値に対するタイムスタンプ

署名が１つしか付けられていなければ、どちらの方式をとっても署名文書のタイムスタンプの効果は同じである。しかし通常は「署名値」に対してタイムスタンプが付けられる。長期署名フォーマットを定めた ETSI のタイムスタンプフォーマット ES-T は「署名値」に対するタイムスタンプと定義している。タイムスタンプは CMS SignedData の SignerInfo にある UnSignedAttributes に署名値に対するタイムスタンプとして置かれる。

5.3.5 複数署名文書へのタイムスタンプの付け方

ビジネス文書では契約書や稟議書などのように複数人の署名が付けられる場合がある。この場合、契約書のように１つの文書に複数人が並列に署名する形式（独立型署名）と、稟議書のように以前に署名された文書に後に署名を重ねていく形式（順序型署名）がある。このような並列署名文書や直列署名文書にどのようにタイムスタンプを付けるのかを以下で考察する。

・並列署名へのタイムスタンプ

並列署名は同一署名対象にそれぞれが署名を付ける（CMS SignedData では SignerInfo を複数指定する）形式である。XML 署名の並列署名も同様に同一署名対象にした<Signature>要素を独立に並列に並べることで実現する。

ここで署名にタイムスタンプを付けるこの場合以下のような疑問が起こる。

a.各署名値にそれぞれタイムスタンプを付けるのか？

b.並列署名を含む署名フォーマット全体に１つのタイムスタンプを付けるのか？

ETSI の ES-T のフォーマットに準じる場合には冗長であっても a.の各署名値にタイムスタンプを付けることになる。CMS SigedData では署名者ごとの SignerInfo を並列に記載することになっており、各 SignerInfo の署名値に対応する UnsignedAttributes にタイムスタンプトークンを付けることになっている。

b.のように署名フォーマット全体に署名を付けた場合、どの署名がいつ生成されたのかが分からなくなってしまう。CMS SigedData には b.の署名フォーマット全体（複数の SignerInfo を含む CMS SignedData）に対して１つのタイムスタンプを付けるフォーマットが用意されていない。したがって並列署名の場合、タイムスタンプは a.のように各「署名値」にタイムスタンプを付けなければならない。

XML 署名にタイムスタンプを付ける場合も ETSI フォーマットを用いれば動揺に各署名値にタイムスタンプを付けたものを<Signature>要素内の<CounterSignature>に置くことになる。

・直列署名へのタイムスタンプ

直列署名の方式は、最初の署名（SignedData の SignerInfo の SignatureValue）を署名対象として署名した SignedData を最初の署名の SignerInfo にある UnsignedAttributes に CounterSignature として置くことで実現できる。CounterSignature は SignerInfo タイプである。次からの署名は直前の CounterSignature の SignatureValue を署名対象にして署名した SignedData を直前の UnsignedAttributes に CounterSignature として置く。順序性は UnsignedAttributes に加える CounterSignature の順序で示す。（最初の CounterSignature はオリジナルの署名値に、次からの CounterSignature は１つ前の CounterSignature の署名値に署名する。）

したがって、直列署名の場合もそれぞれの SigendData の SinerInfo の「署名値」に対応する UnsignedAttributes にタイムスタンプトークンを付けることになる。

XML 署名への直列署名も ETSI TS 101 903 で定めた<UnsignedPropaties>の<CounterSignature>を用いて同様に実現できる。

以上から複数署名にタイムスタンプをつける場合、署名文書全体にタイムスタンプを付けるのではなく冗長に見えても各署名が何時生成されたのかを示すために「署名値」にタイムスタンプを付けなければならない。

5.3.6 CMS SignedAttribute に付ける署名時間

CMS SignedData では SignedAttribute に署名者の署名時刻を付けることができるとしている。この署名時刻はセキュアタイムスタンプではなく署名者がローカルマシンのクロックを使って付ける時刻である。この時刻は正確な時刻である保証は無い。とりあえず署名者が主張する署名時間としての目安である。この時刻と署名検証時に付けたタイムスタンプの時刻が大幅に違っていたり、署名時刻がタイムスタンプ時刻よりも著しく後になっている場合、検証者は署名者が何らかの意図を持って署名したものであってこの署名の有効性を拒否することも考えられる。このような問題を処理するために署名時刻とタイムスタンプ時刻の許容できる差は署名ポリシに記述しておくこともできる。

ETSI の XML 長期署名フォーマットでも<SignedPropaties>の<SigningTime>に署名者が指定する時間を指定する。このデータは署名対象にマージして署名されるが、この時間はセキュアなタイムスタンプではない。

5.3.7 遡った日付でのタイムスタンプ

実世界の契約書の署名日付などでは、契約行為が遅れて合意したので、署名日付を実際に署名した日付ではなく遡った日付を付ける場合がある。TSA が返すタイムスタンプトークンでは TSA の時刻精度の範囲で時刻が記載されるので実世界で行う遡った日付のタイムスタンプは不可能である。（このような要求に対しては別途 out-of-band の方法で処理するしかないであろう？。）

5.4 署名、タイムスタンプトークンおよび長期署名フォーマットの検証

本節では署名の検証（タイムスタンプトークンの検証も基本的には署名の検証と同じである）について基本的操作を確認し、特に長期署名のフォーマットについての検証上の留意点について考察する。

5.4.1 署名の検証

署名の検証は署名者の公開鍵で署名値を復号化し取出したハッシュ値とオリジナルの文書から生成したハッシュ値を比較することで行う。

この際、公開鍵の真正性を証明している証明書の有効性を検証しなければならない。そのためにこの証明書が失効していないかどうかを CRL または OCSP レスポンスから調べる必要がある。署名者の証明書は検証者が信頼する信頼点（トラストアンカー：自己署名証明書を有する Root CA）の証明書からの認証パスで連結されている。署名者の証明書を検証するためには認証パス上にあるすべての証明書とその失効情報（CRL または ARL）を調べることになる。

ここで重要なのは、検証者は信頼点となる CA の証明書（または公開鍵）を確実でかつ信頼できる方法で入手しておかなければならないことである。信頼点の証明書は自己署名証明書を持つ Root CA の証明書である（必ずしも階層の Top の証明書とは限らない）。検証者は信頼点となる CA 証明書のハッシュ値または公開鍵のハッシュ値を out-of-band で入手しておき、認証パス検証時のアンカーに用いる。

・証明書の有効性検証

証明書の有効性検証には以下の２つの手順が必要である。

1） 認証パスの構築：最終的に検証の対象である署名者の証明書から信頼点の CA 証明書までの認証パス（証明書チェーン）を構築すること。

2） 認証パスの検証：構築した認証パスの信頼点の CA 証明書から始めて、すべての認証パス上の中間 CA 証明書と最終の EE 証明書の各種制約を調べ、さらに証明書の失効情報をチェックし、それぞれの証明書の有効性をすべて検証する

ここで認証パスの構築は階層型の CA では単純であるが、メッシュ構造の相互認証を行っている場合は複雑である。特に複雑なメッシュ構造の信頼モデルの場合、認証パス構築は EE 証明書から出発し信頼点に到達するまであらゆる可能性のあるパスを試行錯誤で探索しなければならないのでコストのかかる処理となる。

認証パスの有効性検証のアルゴリズムは RFC3280 の 6 節に詳しく定義されており、検証システムはこのアルゴリズムの処理結果と同じ結果を出すものでなければならない。検証の手順の概要は以下のようになる。

署名者の証明書を入手（通常は署名文書に添付されてくる）。

1） 認証パスの構築

加入者の証明書から出発して検証者の信頼点である CA の証明書までの認証パスをリポジトリなどを検索して証明書チェーン構築する

2） 認証パスの有効性検証

検証者の信頼点である CA の証明書から出発して、認証パス上のすべての証明書の有効性検証を行う

- パス上のすべての証明書の署名が正しいかの検査（証明書発行者の公開鍵で証明書署名を検証する）
- パス上のすべての証明書について有効期限が過ぎていないかの検査
- パス上のすべての証明書について ARL、CRL を検査し、証明書が失効されていないかの検査
- パス上のすべての証明書について証明書拡張のポリシーの一致や各種制約条件を満たしているかの検査

以上のように、証明書の全ての検証がなされた場合にはじめて、署名者の証明書の有効性が確認できる。すなわち、署名名義人の公開鍵が正当なものと判断できる。

・署名フォーマットの検証

署名検証は有効性を確認した公開鍵で行う。プリミティブ署名の検証は指定された署名アルゴリズムで上記手順による検証だけで良いが、一般の署名は CMS SigendData または XML 署名のフォーマットで行われる。これらの署名フォーマットにはそれぞれの構文規定に従って署名が付けられている。したがって署名値の検証の前に CMS SigendData または XML 署名の構文検証を行わなければならない。たとえ署名値自身の検証が正しくとも構文エラーを起こした署名フォーマットの場合の署名は無効となる。

CMS SigendData フォーマットの場合、署名者証明書（認証パスを含む）の検証を行い、署名者の正しい公開鍵を用いて以下の手順による検証を行う。

SignerInfo に signedAttributes が存在しない場合は、EncapsulatedContentInfo が指定している eContent の内容のハッシュ値に対する署名検証を行う。

また SignerInfo に signedAttributes が存在する場合は、(そこには eContent の ContentType と MessageDigest の２つの Attribute が入っているので)、signedAttributes の内容のハッシュ値に対する署名の検証を行う。

SignerInfo が複数あれば（並列署名）それぞれの SignerInfo について同様な署名検証を行う。

それぞれの SignerInfo に UnsignedAttributes があり CounterSignature やタイムスタンプトークンが付けられていれば、CounterSignature の署名検証やタイムスタンプトークンの TSA 署名検証も行わなければならない。

XML 署名の検証の場合は、<SignedInfo>の指定する正規化処理や<Transform>処理を行った後に、<KeyInfo>の署名者の公開鍵の有効性を検証（認証パスの検証を含む）し、<Reference>の指定するデータ（複数あればそれらを連接して）のハッシュ値に対する署名検証を行うことになる。

5.4.2 タイムスタンプトークンの検証

RFC3126 タイムスタンプ・トークンは TSA の応答の TimeStampInfo の部分が CMS SigendData の署名フォーマットの形式に収められている。従って RFC3126 タイムスタンプ・トークンの検証は、

１． TSA 証明書の認証パスの検証
２． タイムスタンプトークンの CMS SignedData 構文の検証
３． タイムスタンププロトコル構文の検証
４． タイムスタンプトークン署名（CMS SignedData の署名値）の検証

が必要になる。

認証パスの検証は証明書の有効性検証の手順をとるが、もし TSA 証明書の信頼点が署名検証者の信頼点と異なる場合は、署名検証ポリシに従って検証者は TSA の信頼点を自分の信頼点とすることに同意し、TSA の信頼点の証明書を信頼できる方法で入手しておかなければならない。

タイムスタンプトークンに時刻認証のための属性証明書が付されている場合にはこの属性証明書の検証も行う必要があるかもしれない。これはタイムスタンプポリシに依存するが、時刻認証

の属性証明書の検証を行う場合は、信頼点 CA…CA→属性認証局の認証パスの検証を行い、有効性が確認された属性認証局の公開鍵で属性証明書の署名の検証を行うことになる。この場合の信頼点は多分 TSA の信頼点と同じであろうが、もしこの信頼点が別物であり、かつ検証者自身の信頼点とも異なる場合、この信頼点を信頼するかどうかは属性認証局と検証者のポリシによる。

XML タイムスタンププロトコルが制定されれば、タイムスタンプトークンの検証は XML 構文の検証を含まなければならない。

5.4.3 長期署名フォーマットの検証

ETSI TS 101 733 の長期署名フォーマットの検証は少し複雑である。それはこのフォーマット構文が複雑であるばかりでなく、ここには対象オブジェクトに対する署名の他に ES-T のタイムスタンプ、ES-X-Long に含まれる証明書や CRL があり、さらに ES-A のタイムスタンプなど、それぞれの構文チェックや署名検証が多数必要になるからである。

ここでは最も複雑な ES-A のフォーマットを検証するときの留意点を考察する。

まず最初に ES の署名ポリシに従って以下のように ES-X-Long のフォーマットの検証を行う。

1. ESの署名ポリシOIDの示す署名ポリシに従ってこの署名フォーマット構文が正しく構成されているかを検査する。
2. 署名ポリシで指定された信頼点を確認し、ES-X-Long に含まれる認証パス上の証明書と失効情報データを使って署名証明書の有効性を確認する。この際 ES-C の各証明書や失効情報の参照値（各証明書の加入者の DN とハッシュ値や失効情報のハッシュ値等）と実際の証明書や失効情報の比較検証を行う。この際認証パス上にあるすべての証明書のついて、証明書の内容である証明書ポリシや鍵仕様目的や各種制約についても検証すべきである。この証明書内容の検査でエラーが発見された場合最初の検証者の署名検証に落ち度があったことになる。

   もし、失効情報に OCSP レスポンスが使われている場合は OCSP レスポンダの認証パスについても検証を行う。
3. 署名証明書の公開鍵を用いて ES の署名値の検証を行う。
4. ES-T のタイムスタンプ認証パスを検証し、CMS SignedData の UnsignedAttributes にあるタイムスタンプの検証を行う。
5. CMS SignedData の UnsignedAttributes に CounterSignature があればその署名の検証も行う。
6. 次に、最初の ES-A のアーカイブタイムスタンプ１つが付いたタイムスタンプの検証の場合は、上記の ES-X-Long フォーマットを検証した後にアーカイブタイムスタンプの検証を行う。これは生きている（有効期限内の）タイムスタンプ署名なので、オンラインで ES-A のタイムスタンプの認証パスを検証し、タイムスタンプサーバの公開鍵でタイムスタンプ署名を検証し、タイムスタンプの時刻等のフォーマットも検証を行うことができる。
7. 超長期のアーカイブで複数のアーカイブタイムスタンプの付いた ES-A フォーマットの場合は上記の ES-X-Long フォーマットを検証した後、既に有効期限を越えている内側にある ES-A のタイムスタンプは認証パスを有していないので認証パスは検証できないが（これに

ついては以下の「アーカイブタイムスタンプの考察」で問題点を検討する)、タイムスタンプ証明書の公開鍵でこのアーカイブタイムスタンプは検証できる。期限の切れた内側のアーカイブタイムスタンプは最も外側の有効な（生きている）アーカイブタイムスタンプで完全性が保証されており、この偽造は検出可能である。最外部のアーカイブタイムスタンプの検証は６．の手順で行う。

アーカイブタイムスタンプについての考察

ES-T のタイムスタンプは署名者または最初の検証者が付け、その署名の認証パスに加えて、タイムスタンプの認証パスもその reference と Complete Data を CMS SignedData の unsigendAttribute に置くことが出来る。(XML フォーマットでは<UnsigendPropaties>におく）

しかし、アーカイブタイムスタンプは、ES-T が署名値に対するタイムスタンプであるのに対して、ES-X Long のすべての構成要素を対象にしてそのハッシュに対するタイムスタンプ(署名）である。この時点で reference と Complete Data も含めてアーカイブタイムスタンプが付けられ、SignedData の unsigendAttribute の最後に置かれる。しかし、このアーカイブタイムスタンプの認証パスを格納する場所が ETSI のフォーマットに定義されていないので置き場所が無い。

アーカイブタイムスタンプはそれが生きている間は自身の認証パスを残さなくともオンラインでタイムスタンプ検証ができる。そこで、アーカイブタイムスタンプの認証パスは残さないとする。さてアーカイブタイムスタンプの有効期限が切れる前に、次のアーカイブタイムスタンプを付けるのですが、内側のアーカイブタイムスタンプが死んでしまった時、再検証者は内側のタイムスタンプの検証をどうするのかと言う問題が生じる。誰かがこの内側のタイムスタンプを偽造することで署名が偽造できてしまうのではないかと言う疑問が生じる。

２番目のアーカイブタイムスタンプを付ける人が信頼できれば､その人は ES-X-L のデータを用いて署名検証を行い、まだ生きている最初のアーカイブタイムスタンプも検証してこのアーカイブタイムスタンプも含めて、2 番目のアーカイブタイムスタンプを付けるはずである。従って、時間が経って最初のアーカイブタイムスタンプが死んでしまっても、最初のアーカイブタイムスタンプを偽造すればその検出が可能である。

ここではアーカイブタイムスタンプをつける人は信頼できる人であることを前提にしている。この前提は妥当なものかもしれない。長期署名を保存しようと言う人は署名の偽造を守るためにアーカイブタイムスタンプを付けるのであって、自分で偽造することは自らを損害に導いてしまうからである。

しかし､このアーカイブタイムスタンプを付ける人が信頼できない場合はどうであろう｡もし、このとき、ES の署名アルゴリズムが危殆化しており、その公開鍵から署名鍵が推定できたとしよう。この場合署名用証明書の認証パスを変えることなく、署名の偽造が可能である。この場合、署名値が変わるので、ES-T のタイムスタンプを付け替えなければならない。しかし勝手なタイムスタンプで過去に遡ってタイムスタンプを偽造したとするとタイムスタンプの認証パスの検証で偽造が発見される。しかし、タイムスタンプの署名アルゴリズムも危殆化していればタイムスタ

ンプの認証パスを変えないでタイムスタンプを差換えることが出来る。

このタイムスタンプは内側のアーカイブタイムスタンプで署名されているので、悪人である２番目のアーカイブタイムスタンプを付ける人はこの内側のタイムスタンプを遡った時刻のアーカイブタイムスタンプに差換えるのである。この差換えはアーカイブタイムスタンプの認証パスとその信頼点が保存されていないので勝手な偽造タイムスタンプが付けることが可能になるからである。そこで再外部の生きているアーカイブを付けるのである。こうして偽造した長期署名フォーマットは第 3 者からは正しいものと認識されてしまう。

以上の考察から ES-T のタイムスタンプはその認証パスに関する参照と完全データは ES-C と ES-X-L に残す。アーカイブタイムスタンプは常に外側のアーカイブタイムスタンプは生きていることとし、アーカイブタイムスタンプの信頼点のセキュアな保管と認証パスを何らかの方法で残しておくべきである。

5.5 タイムスタンプポリシ

タイムスタンプのポリシは運用ガイドで詳しく述べるので、ここでは利用上の留意点としてポリシに関連する項目について述べる。

TSP（RFC3161）にはタイムスタンプ要求に付いてはポリシ指定は任意だが、タイムスタンプトークンにはタイムスタンプポリシーが必須項目として含まれる（TIML でも同様)。ポリシには運用上の観点と鍵管理の実施方法等の規定の他に精度と順序性（オーダリング）の規定を示すことになっている。ここでは簡単にこれらの留意点について述べておく。

5.5.1 時刻およびその精度

タイムスタンプトークンに示された時刻はどのようにして信頼できるものであろうか？タイムスタンプトークンに示される時刻およびその精度は TSA 利用者の重要な関心事項である。利用者は TSA がどのように時刻を信頼できるように運用されているかについて関心を持たなければならない。たとえ TSA が GPS からの時刻や電波時計を用いていると言っていてもそれを検証する手段が無い。利用者はトークンに示された時間精度を単に信用するだけではなく時刻がどのように信頼できるかについても検証できる手段が求められるようになろう。利用者は TSA が標準時間とどのように校正しているかについて TSA の用いる時刻認証の方法が求めるようになろう。

時刻認証については現在それを TSA と利用者に知らせ検証を可能にする方法の標準はまだ定められていないが、属性証明書を使って時刻認証を提供する方法が Datum 社（セイコークロノトラスト）で実装している。

5.5.2 オーダリング（順序）

タイムスタンプトークンで指定されるオーダリングはそのフラグが False の場合は、最初のトークンの時刻（genTime）と次のトークンの時刻の差が２つのトークンの精度の合計より大きいことを意味する。もしフラグが True の場合は、すべてのトークンが精度に関係無く時刻（genTime）の順序で発行されることを意味する。

TSA がオーダリングを維持するのは通常難しく多くの TSA でもオーダリングのフラグは False としている。利用者はこのことに注意してトークンを受理しなければならない。

5.5.3 タイムスタンプ署名鍵のバックアップ

TSA はトークン署名鍵を鍵の HSM の破損などの事態に備えてバックアップすべきであろうか？TSA の運用では TSA はトークン署名鍵をバックアップする必要はないし、すべきではない。もし TSA 署名鍵が漏洩したり危殆化したのではなく事故によって利用できなくなったのであれば、TSA 今までの証明書を失効させず、TSA は新しい署名鍵を生成し以後新しい TSA 証明書を用いれば良いからである。以前のタイムスタンプトークンは有効である以前の証明書で検証できるからである。TSA は PKI として見た場合、Time Stamp Authority と言っているが、CA とは違い End Entity の立場にある。TSA の署名鍵は TSA の利用者の信頼点ではない。つまりタイムスタンプトークンの署名は、通常の CA の Subscriber としての End Entity がドキュメントにする署名と同じである。バックアップは否認防止を阻害する要因になりうる。バックアップ鍵が存在するだけでも署名は本来の署名者によって成されたのではなく、バックアップ鍵にアクセスした誰かが署名したと主張できるかもしれないからである。これを避けるためにはセキュアなバックアップ鍵の保管管理が必要で TSA 運用コストもかさむことになる。

5.5.4 タイムスタンプ署名鍵（TSA 署名鍵）の安全性

タイムスタンプの署名鍵はどの程度の強度を持てばよいであろうか？タイムスタンプは公証機能を持っているので、TSA 証明書が短い有効期間ではその効用が限られる。RSA 換算で最低 1024 ビットの鍵長は必要である。長期署名フォーマットではもしアーカイブ目的の ES-A のタイムスタンプに用いるのであれば ES-T に用いるタイムスタンプの鍵長より強度の高い 2048 ビットの CA と同程度の強度を持つ鍵長を用い 10 年以上の有効期限を持つことが望ましい。

5.5.5 TSA 証明書の発行周期

TSA 証明書がその有効期限が残り少なくなった時点で発行されたタイムスタンプを用いた場合、タイムスタンプ自身の有効期間が短くなり、その可用性が少なくなる。従って TSA は TSA 証明書の有効期限の半分を経過した時点で次の TSA 証明書を要求し以後タイムスタンプはこの新しい証明書をもちいて行うことが望ましい。このようにすれば最も有効性が短いタイムスタンプでも TSA 証明書の有効期限の半分は保障されることになる。

5.5.6 タイムスタンプトークンとタイムスタンプ対象データとのリンク

タイムスタンプ要求者は取得したタイムスタンプトークンとタイムスタンプ要求時に用いたデータのハッシュ値との関連を明確にして保存しておかなければならない。事後の検証を曖昧にしないために必要である。このことはデジタル署名の長期保存に関しては、標準化された長期署名フォーマットを使うことで明確にできる。ASN.1 の長期署名フォーマット（ETSI TS 101733）あるいは XML 長期署名フォーマット（ETSI TS 101 903）ではオリジナルの署名対象の署名値を明確に指定し、この署名値にタイムスタンプを付けることを定めている（ES、XAdES）。また Archive Time

Stamp についてもこれらの仕様でタイムスタンプ対象を明確に規定している。したがって、標準の長期署名フォーマットを使えばタイムスタンプ対象のデータとこれに対するタイムスタンプトークンとの対応は明確になっている。

6．付録

6.1 用語集

本ガイドラインで使用される主要な用語の一部を、以下に解説する。

1．タイムスタンプ要求者：
　タイムスタンプクライアントを用いてタイムスタンプ発行者にタイムスタンプされることを望むデータに対するタイムスタンプを要求し、タイムスタンプの発行を受ける者。

2．タイムスタンプ検証者：
　タイムスタンプクライアントを用いて所有するデータに対するタイムスタンプの有効性を検証する者。あるいは、タイムスンタプクライアントを用いてタイムスタンプ発行者にタイムスタンプの検証を要求し、検証結果の発行を受ける者。

3．タイムスタンプ局（TSA：Time-Stamping Authority）：
　タイムスタンプサーバを用いてタイムスタンプを発行する機関。タイムスタンプの検証要求を受け、検証結果を発行する場合もある。また、発行した複数のタイムスタンプから生成したハッシュ値等の証拠情報を新聞や定期刊行物などの公開メディアに掲載する場合もある。

4．時刻配信局（TA：Time Authority）：
　専用の装置やソフトを用いて、時刻配信、時刻監査、時刻認証を実施する信頼された第三者機関。時刻監査証明書を発行することもある。

5．時刻配信：
　タイムスタンプサーバに標準時刻を配信すること。

6．時刻監査：
　タイムスタンプサーバを特定し、その時計の標準時刻との差異を計測すること。

7．時刻認証：
　タイムスタンプサーバを特定し、その時計の標準時刻との差異を計測し、差異が存在した場合はその時計を調整して標準時刻と同期をとること。

8．時刻監査証明書：
　時刻監査を行った事実やその結果､あるいは時刻認証を行った事実や内容を証明する証明書。

9．国家標準時配信機関（NTA ： National Time Authority）：
国家の標準時刻を生成し、配信する信頼された機関。

10．監査人：
タイムスタンプ局のログや証拠情報などを検証することにより、信頼に足るタイムスタンプ発行がなされているか否かを検証する者。

11．認証局：
証明書の発行、開示、失効もしくは一時失効等のサービスを行う信頼された第三者機関。

12．リポジトリ：
公開鍵証明書や証明書失効リストなどを保管し、証明書利用者等に対してこれらの開示や配布もしくは検索等のサービスを提供するシステム。

13．署名ポリシ発行局：
署名ポリシの参照情報（OID など）と署名ポリシの内容を登録／管理し、電子署名文書内に示された署名ポリシの参照情報をキーとした問い合わせに応え、署名ポリシ本体情報を発行する機関。

14．TTP（Trusted Third Party）：
情報セキュリティ関連業務の管理・運用状況について他のエンティティから高い信頼を寄せられているエンティティ。

15．耐タンパ性：
装置を分解するなどして、中にある秘密情報を不正に入手あるいは改竄しようとする行為（Tamper）に対する耐性。

16．ASN.1 ：
ISO/ITU-T で規定した抽象構文表記法､OSI で定めた 7 層通信プロトコルの 6 層プレゼンテーション層の規約である｡データ構造を表現させるためにタグによってデータの意味付けを行う。この構文をエンコードするとバイナリデータとなる。エンコード法には各タグ内でデータ長を規定しない BER とデータ長を指定する DER がある。

17．CMS SignedData ：
IETF S/MIME WG で定めた S/MIME のための署名データタイプ｡この署名フォーマットは S/MIME だけでなく一般の署名フォーマットとして使われる。ASN.1 でエンコードされる。

18. CRL :

Certificate Revocation List、失効リストと言われ、公開鍵証明書（X.509）の失効した証明書のリスト。通常リポジトリに公開され、証明書の検証者が証明書が失効していないかどうかを検索してし調べるために用いる。

19. Counter Signature :

対向署名、署名に対して順序付けた多重署名として直列に署名する署名

20. Detached 署名 :

XML 署名の形式の１つ。署名要素の外部のファイルに対しての署名対象を指定する方式。

21. Enveroped 署名 :

XML の文書の内部に署名要素を含む XML 署名形式。

22. Enveroping 署名 :

署名要素の中に署名対象を含む署名形式。

23. OCSP (Online Certificate Status Protocol) :

オンラインで証明書の状態を問合わせるためのプロトコル。OCSP レスポンダは問合わせに対し、証明書の状態を Valid、Invalid、Unknown の応答を行う。

24. PKCS#1 :

RSA 方式の署名や鍵の暗号を行うためのフォーマット。

25. RSA-PSS :

Probabilistic Signature Scheme、PKCS#1 の v1.5 では署名野安全性に付いて証明できないが、PSS 方式では一定の仮定の基に確率的に署名の正当性を証明できる方式。

26. SignedAttribute :

CMS SigendData の署名フォーマットで署名対象の文書に加えて署名する署名属性。

27. TIML (Temporal Integrity Markup Language) :

XML 構文で規定したタイムスタンプの構文。OASIS の DSS(Digital Signature Service)の TC で策定中のタイムスタンプ標準（現在ドラフト）

28. TSP :

タイムスタンププロトコル。ASN.1 方式の RFC3161 と XML の TIML がある。

29. TimeStamp Token :
TSA が発行するタイムスタンプ

30. UnsignedAttribute :
非署名属性、CMS SignedData で署名者の署名をしない属性、タイムスタンプや Counter Signature がその例である。

31. アーカイブタイムスタンプ :
長期にデータを保存するとき署名やタイムスタンプの寿命を延長するために用いるタイムスタンプ。

32. タイムスタンプポリシ :
タイムスタンプトークンが共通のセキュリティ要求を持つ特定の利用者グループやアプリケーションのクラスへの適用性を示す名前付けした規則。
TSA の運用規定としての義務や責任およびセキュリティ対策を示し、使用する時刻の信頼性に付いての方針を述べたもの。

33. 署名ポリシ :
第三者の署名ポリシ発行者が作成する署名者と検証者が合意する署名に関する一定の規則。署名方法や信頼点の指定などが定められる。

34. 正規化処理 :
Canonicalization、正規化、XML 記述を曖昧さ無しに整形する方法。

35. 直列署名 :
Counter Signature として前に署名に対して署名する順序付けた署名。ワークフローなどの署名に用いられる。

36. 認証パス :
エンドエンティティの証明書からの発行 CA 証明書のチェーンで信頼点までのパス。信頼点をアンカーにして最終のエンドエンティティ証明書の有効性を検証するのに用いられる。

37. 並列署名 :
1 つの文書に対して複数の署名者が並列に署名する形式。契約書の署名などに用いられる。

6.2 参考文献

今回の検討を進める過程で参考とした文献を以下に示す。特にＷＥＢ上の文書はバージョンが更新されることがあるので、注意を要する。

１．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」、日本銀行金融研究所、2000 年４月

２.RFC3161,“Internet X.509 Public Key Infrastructure: Time-Stamp Protocol (TSP)”,2001.8

３．RFC3126,“Electronic Signature Formats for long term electronic signatures”,2001.9

４．ETSI TS 101 903,“XML Advanced Electronic Signatures (XAdES)”,2002.2

５．RFC3125,“Electronic Signature Policies”,2001.9

６．ETSI TR 102 038,“XML format for signature policies”,2002.4

７．NEMA Standards Publication PS 3 Supplement 41“DICOM Digital Signatures”

８．IETF RFC3161bis,“Internet X.509 Public Key Infrastructure: Time-Stamp Protocol (TSP)”

９．ISO/IEC 18014-2：2002,“: Information technology? Security techniques ? Time stamping services ? Part 2: Mechanisms producing independent tokens”

10．ISO/IEC 18014-3：2002,“Information technology ? Security techniques ? Time stamping services ? Part 3: Mechanisms producing linked tokens”

11．ETSI TS 101 733,“Electronic Signature Formats” v 1.4.0, 2002.4

12．電子商取引推進協議会、認証・公証ワーキンググループ、「電子署名文書長期保存に関する中間報告」、H13－認証・公証 WG－3、2001 年 3 月

13．電子商取引推進協議会、認証・公証ワーキンググループ、「電子署名文書長期保存に関するガイドライン」、H14－認証・公証 WG－3、2002 年 3 月

14. ETSI TS 102 023：“Policy requirements for Time-Stamping Authorities”,2003.1

15. RFC2630, “Cryptographic Message Syntax”,1999.6

16. RFC3369, “Cryptographic Message Syntax”(CMS),2002.8

17. RFC3370, “Cryptographic Message Syntax (CMS) Algorithms”,2002.8

18. RFC3275, “(Extensible Markup Language) XML-Signature Syntax and Processing”,2002.3

19. RFC3076, “Canonical XML Version 1.0”,2001.3

20. RFC2510, “Internet X.509 Public Key Infrastructure: Certificate Management Protocols, 1999.3

21. OASIS DSS TC, “Tokens and Protocol for the Temporal Integrity Markup Language (TIML)”, Working Draft 01, 6 September 2002
http://www.oasis-open.org/committees/dss/

22. RFC3280, “Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile”,2002.4

23. RFC3447 Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1, 2003.2

24. RFC2315 PKCS #7: Cryptographic Message Syntax Version 1.5, 1998.3

25. ETSI TR 102 041 Signature Policies Report, 2002.2

RFC は以下の URL の RFC 番号で検索できる
http://www.ietf.org/rfc.html

ETSI は以下の URL で参照できる
http://portal.etsi.org/esi/el-sign.asp

メンバーリスト

事務局

川松　和成　電子商取引推進協議会　主席研究員
松山　博美　電子商取引推進協議会　主席研究員
前田　陽二　電子商取引推進協議会　主席研究員

顧問

松本　　勉　横浜国立大学大学院
平田　健治　大阪大学大学院

リーダー

木村　道弘　日本電気株式会社
宮崎　一哉　三菱電機株式会社
櫻井　徹　　株式会社 NTT データ

TF5 メンバー（編集メンバー）

氏名	会社名
鈴木　邦康	株式会社ＮＴＴデータ
磐城　洋介	ＮＴＴコムウェア株式会社
野村　　進	ＮＴＴコミュニケーションズ株式会社
鈴木　優一	エントラストジャパン株式会社
上畑　正和	セイコーインスツルメンツ株式会社
雨宮　隆征	セイコーインスツルメンツ株式会社
秋山　将	日本電信電話株式会社
島　成佳	日本電気株式会社
近藤　弓末	ソニー株式会社

SWG3 メンバー（参加メンバー）

氏名	会社名
河田　悦生	株式会社エヌ・ティ・ティ・ドコモ
関野　公彦 ＊	株式会社エヌ・ティ・ティ・ドコモ
風間　博之	株式会社ＮＴＴデータ
宍倉　勝仁	シャチハタ株式会社
岩崎　善徳 ＊	セイコーインスツルメンツ株式会社
藤川　真樹	綜合警備保障株式会社
星野　　理	株式会社帝国データバンク
藤岡　直美	日本アビオニクス株式会社
小暮　貢次郎	日本信販株式会社
野口　雄治	日本認証サービス株式会社
浅野　昌和	日本ボルチモアテクノロジー株式会社
松永　和男	株式会社日立製作所
永倉　　俊	富士通株式会社
小谷　誠剛	富士通株式会社
西谷　研次	株式会社 UFJ 銀行

（注）＊はオブザーバー

平成 14 年度
ＥＣ技術基盤の相互運用性に関する調査研究事業
（電子署名生成・検証システムのセキュリティ環境の
国際標準化等の調査）
タイムスタンプサービスの利用ガイドライン
平成 15 年３月発行

発行所　財団法人　日本情報処理開発協会
電子商取引推進センター
東京都港区芝公園３丁目５番８号
機械振興会館３階
TEL：03(3436)7500
印刷所　新高速印刷株式会社
東京都港区新橋 5-8-4
TEL：03(3437)6365

この資料は再生紙を使用しています。