オフライン型タイムスタンプサービスの設計および TSA とクライアントの実装

掛井 将平[†,‡] 脇田 知彦[‡] 毛利 公美[†] 白石 善明[‡] 野口 亮司[††]

†岐阜大学
501-1193 岐阜市柳戸 1-1

‡名古屋工業大学
466-8555 名古屋市昭和区御器所町

††株式会社豊通シスコム
450-0002 名古屋市中村区名駅 4-5-28 近鉄新名古屋ビル

あらまし タイムスタンプサービスとは，電子データがある時刻において存在し，それ以降改ざんされていないことを TSA（Time Stamp Authority）により保証するサービスである．我々は既に，規模拡張性／頑強性／外部秘匿性を持つ階層型タイムスタンプサービスの提案を行い，そのモデルの安全性について議論した．本稿では，提案モデルに基づいたオフライン型タイムスタンプサービスについて述べる．まず，サービスのシステム構成を示し，サービスを構成する主体の機能要件を定義した．そして，REST（Representational State Transfer）アーキテクチャスタイルによる TSA サービスの実装とクライアントにおける TPM を用いた安全なタイムスタンプの生成について述べる．TPM を用いることにより提案モデルの安全性に基づいたオフライン型のシステムを実装できることを示している．

Design of Offline Time-Stamp Service and Implementation of Its TSA and Client

Shohei Kakei[†,‡] Tomohiko Wakita[‡] Masami Mohri[†] Yoshiaki Shiraishi[‡] Ryoji Noguchi[††]

†Gifu University
1-1 Yanagido, Gifu 501-1193 JAPAN

‡Nagoya Institute of Technology
Gokiso-Cho, Showa-Ku, Nagoya 466-8555 JAPAN

††Toyotsu Syscom Corporation
4-5-28 Meieki Nakamura-Ku, Nagoya, 450-0002 JAPAN

Abstract Time Stamp Service provides verifiability of ‘ proof of existence’ and ‘ proof of completeness’. We have proposed the Hierarchical Time Stamp Service with scalability, robustness and confidentiality, and showed the security evaluation results of the proposed model. This paper discusses Offline Time-Stamp Service based on the proposed model. We show the design of system architecture, and functional requirements of each entity of the service. We implement the TSA service based on REST style architecture and secure issuing of time-stamp using TPM.

1. はじめに

電子データに信頼できる時刻を付与する仕組みに時刻認証がある．時刻認証とは，電子データがある時刻に存在し，それ以降改ざんされていないことを証明するものである．

電子データと信頼できる時刻を関連付けるためにタイムスタンプ（TS）が利用される．TS は TSA（Time Stamp Authority）により発行され，電子データのハッシュ値，TSA により付与された信頼できる時刻，TSA の署名が含まれている．TS に求められる要件について文献[1]で次のように示されている．

[電子データの存在証明]
TS を付与した電子データがある時刻以前に存在していたこと，あるいは他の電子データとの順序関係を証明することができる．

[電子データの完全性証明]
TS を付与した時点から電子データが改ざんされた場合にこれを検出できる機能を持つ．

これらの要件を満たす電子署名方式やリンキング方式などのタイムスタンプ技術が存在する[1].

日本の商用のタイムスタンプサービスでは電子署名方式[3][4][5][6]が普及している．電子署名方式は RFC3161 [2]において PKI TSP として標準化されている．PKI TSP では，外部にある単一の TSA が TS の生成／発行を行う．この方式では，TS の発行処理速度は TSA の処理能力やネットワークの帯域に依存する．災害時などの TSA との通信が途絶する場合， TS を要求することができない．また，TS を要求した事実を外部に秘匿できない．

我々は，[7]においてこれらに対応するために規模拡張性／頑強性／外部秘匿性を持った階層型タイムスタンプサービスモデルを提案している．そして，提案モデルの安全性評価を行い，安全に提案モデルを実現するための方法を示している．

本稿では，提案モデルの安全性の評価結果をもとに階層型タイムスタンプサービスモデルに基づくオフライン型タイムスタンプサービスを実現するための TSA サービスとクライアントの実装を行う．オフライン型では，TS 要求者と TS 発行者が同一の端末内に存在するので，安全な TS の生成が課題となる．そこで，セキュリティチップである TPM（Trusted Platform Module）を利用することを考え，提案システムが[7]で示した安全性を満たし

ていることを示す.

2. 階層型タイムスンプサービス

我々は[7]において，規模拡張性／頑強性／外部秘匿性を持つ階層型タイムスタンプサービスモデルの提案を行い，その安全性の評価を行った．以下ではその概要を述べる．

2.1. モデルの概要

階層型タイムスタンプサービスモデルを図 1 に示す.

2.1.1. エンティティ

[ATC : Absolute Time Certifier] 絶対時刻認証者．サービス提供者が用意するエンティティで，内部時計は信頼できる時刻源（UTC など）と同期している．RTC に対して，時刻認証権限を委譲することにより，時刻認証処理の負荷分散を行う.

[RTC : Relative Time Certifier] 相対時刻認証者．ATC から委譲された時刻認証権限により，Client に対して時刻認証処理を行う．時間間隔が保証された内部時計を持っている.

[Client] 時刻認証要求者．RTC に時刻認証要求を行うことによって，時刻認証を受ける.

[Verifier] 検証者. ATC の時刻認証権限委譲処理, RTC の時刻認証処理が適切に行われたことを確認する.

2.1.2. トランザクション

[時刻認証権限委譲処理] ATC が, RTC の内部時計の時刻情報に対して時刻認証を行う．つまり，ATC が発行する TS を用いて，ATC と RTC の内部時計を関連付けることにより，時刻認証権限を委譲する.

[時刻認証処理] RTC は自身の内部時計より，時刻認証権限委譲処理からの経過時間を計測する．経過時間と ATC が権限委譲時に認証した時刻から現在の時刻を算出する．この時刻を用いて，Client に対して TS を発行し，時刻認証を行う.

[検証処理] 時刻認証権限委譲処理／時刻認証処理により発行された一連の TS の検証用データを各エンティティから収集．そのデータをもとに TS の完全性を検証する.

2.2. 安全性評価の概要

攻撃者は，Client の一人とし，ATC／RTC が発行する TS（ATS／RTS とする）の改ざんを行う．また，攻撃者

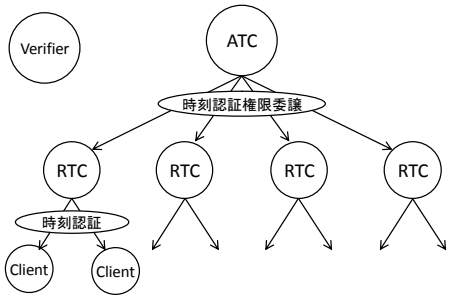

図 1 階層型タイムスタンプサービスモデル

表 1 攻撃者の攻撃パターン

	攻撃者がRTCと結託	攻撃者がRTCになりすます	攻撃者はRTCと結託／なりすましを行わない
攻撃者がATCと結託	P1	P2	P3
攻撃者がATCになりすます	P4	P5	P6
攻撃者はATCと結託／なりすましを行わない	P7	P8	P9

α ：電子署名を生成するための暗号技術の安全性が低下する場合

β ：電子署名を生成するための暗号技術の安全性が低下しない場合

表 2 階層型タイムスタンプサービスモデルの安全性評価結果

パターン		P1	P2	P3	P4	P5	P6	P7	P8	P9
α	ATS	×	×	×	×	×	×	○	○	○
	RTS	×	×	○	×	×	○	×	×	○
β	ATS	×	×	×	○	○	○	○	○	○
	RTS	×	○	○	×	○	○	×	○	○

○･･･TSの改ざんを検知できる
×･･･TSの改ざんを検知できない

の攻撃パターンを表 1 に示す.

安全性評価の結果を表 2 に示す．Verifier が TS の改ざんを検知できるときが○, TS の改ざんを検知できないときが×である．表 2 の色付きの部分に該当するパターンは，タイムスタンプサービス提供者の ATC と攻撃者が結託を行う可能性は低く，また，暗号技術の安全性が低下する前に何らかの対応をとるという状況を想定した場合である．TS を安全に生成し，時刻認証権限委譲処理／時刻認証処理を安全に行うためには，Verifier が β-P4, β-P7 において RTS の改ざんを検知できれば，現実的な運用の中で安全な階層型タイムスタンプサービスモデルを実現できる.

β-P4, β-P7 における RTS の改ざんを防ぐためには, RTC の運用方法を考慮する必要がある．専用サーバで RTC を運用する場合, 適切なサーバ管理者により攻撃を防ぐことができる．しかし，本稿ではオフライン型タイムスタンプシステムの実現を目的として, 端末内で RTC を運用するため，適切な管理者を配置できないと考え，セキュリティチップにより RTC の安全性を確保する．次章では，オフライン型タイムスタンプサービスのモデルとシステムについて述べる.

3. オフライン型タイムスタンプサービス

本サービスでは，端末内にある RTC が時刻認証処理を行うため，外部すなわち ATC との通信を必要最小限に抑えることができる． つまり, 時刻認証処理にネットワークの帯域や TSA の処理能力を考慮する必要がなく，大量の電子データの時刻認証が可能となる.

3.1. サービスモデル

オフライン型タイムスタンプサービスモデルを図 2 に示す. RTC と Client を同じ端末内で動かすことによって，時刻認証処理をオフラインで行う.

3.2. サービスの要件

本サービスでは，オフラインで TS が生成／発行され

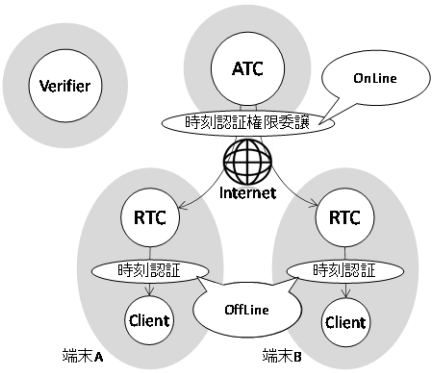

図 2　オフライン型タイムスタンプサービスモデル

るため，オンラインで処理を行う既存のタイムスタンプサービスと状況が異なる．そこで，RTC によりオフラインで発行された TS は，1 章で示した要件を拡張した次のものを満たしている必要がある．

- オフラインで発行した TS により電子データの存在証明ができること
- オフラインで発行した TS により電子データの完全性証明ができること

これらの要件を満たすためには，時刻認証権限委譲処理／時刻認証処理が適切に行われている必要がある．そして，それらの処理は 2.2 節から攻撃者と ATC／RTC が結託してはならないことが分かっている．そこで，ATC／RTC はそれぞれ攻撃者からの攻撃を防ぐ手段を講じる必要がある．本サービスでは，ATC の適切なセキュリティ管理や RTC のセキュリティチップの利用を行う．

3.3. サービスのシステム構成

オフライン型タイムスタンプサービスのシステムの動作を図 3 に示す．本サービスでは，同じ端末内で TS の要求／生成／発行が行われる．そこで，RTC は TS を安全に生成するためにセキュリティチップを利用する．

サブシステム構成を図 4，図 5，図 6 に示す．ATC は，認証に成功した RTC にのみ時刻認証権限を委譲する．RTC は，Client の TS 要求に応じて，TS の生成／発行を行う．Verifier は検証対象の Client からの要求に応じて，Client が持つ TS を ATC／RTC が持つ TS 検証用データにより検証する．Client と RTC はオフラインで接続されており，それ以外はオンラインで接続されている．

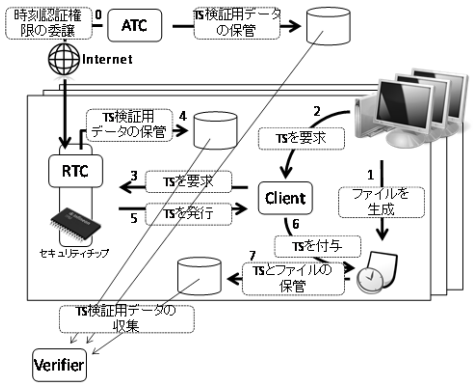

図 3 オフライン型タイムスタンプサービスのシステムの動作

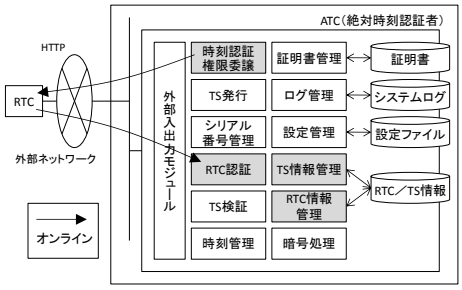

図 4　ATC の構成

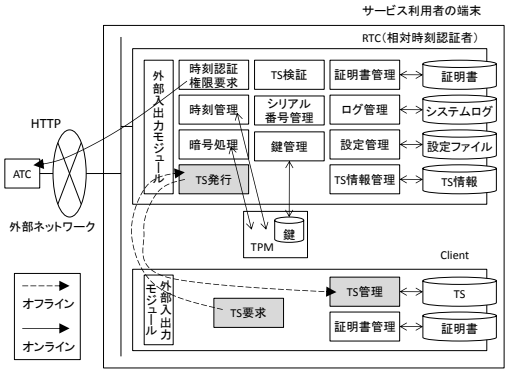

図 5　RTC／Client の構成

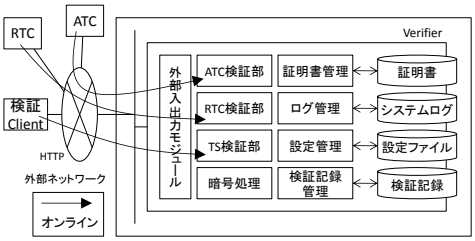

図 6　Verifier の構成

3.4. サービスを構成する主体の機能要件

3.2 節で挙げたサービスの要件を達成するために必要な機能要件を表 3 に示す．3.2 節で挙げたように，サービスの要件を満たすためには時刻認証権限委譲処理／時刻認証処理が適切に行われなければならない．そこで，ATC による権限委譲先の RTC の認証が必要となる．また，ATC は認証済みの RTC に対して権限委譲を行う必要がある．そこで，システムのスケーラビリティ向上を目的とした REST スタイルアーキテクチャによる ATC の実装も必要となる．また，RTC は時刻認証処理を安全に行うために TPM を用いる．

他にも，既に存在する TSA を ATC として使うことを考えれば，RFC3161 準拠のフォーマットで TS の要求／生成を行うことが必要となる．

表 3　システムの各主体の機能要件

主体	機能要件	要件を満たすための手段
ATC	RTCを認証できること	TPMのベンダーを特定するための情報を利用（EK証明書など）
	RTCの時刻保証権限委譲要求に応じて，権限委譲できること	RESTアーキテクチャスタイルで実装
RTC	ATCに時刻保証権限委譲を要求して，委譲を受けることができること	RFC3161準拠のフォーマット／手順で発行
	ATCと接続していなくても，Clientからのタイムスタンプ発行要求に応じて，発行できること	RFC3161準拠のフォーマット／手順で発行
	Clientに発行するタイムスタンプを安全に出力できること	TPMのTickCounter機能を利用
Client	RTCにタイムスタンプを要求して，取得できること	RFC3161準拠のフォーマットで要求
Verifier	Clientから検証依頼のあった，オフラインで発行されたタイムスタンプを検証できること	TSの電子署名を検証

4.　オフライン型タイムスタンプサービスのシステム実装

本章では，3.4 節で挙げた機能要件の中でも，「REST による ATC の実装」と「TPM による安全な TS の生成処理の実装」について説明する．

4.1. 提案システムの開発環境

端末内で RTC と Client を動かし，Web サーバとして ATC を動かす．端末にはバージョン 1.2 の TPM が搭載されている．開発言語は Java を使用し，端末／Web サーバとも JRE1.6.0_21 がインストールされている．通信部分は HTTP を用いている．TS 関連の処理には IAIK TSP2.1[8]，TPM 関連の処理には IAIK jTSS[9]を使用している．

4.2. REST アーキテクチャスタイルによる ATC の実装

本節では，オフライン型タイムスタンプサービスを実装する際に，既存のタイムスタンプサービスと異なる点について述べる．具体的には ATC から RTC への時刻認証権限委譲処理の実装に関する部分である．図 4 の色付きが関連モジュールである．

4.2.1. REST アーキテクチャスタイルによる Web サービスの構築

REST で Web サービスを構築するには，まず，サーバのリソースを一意に特定するために URI を割り当てる．そして，そのリソースを HTTP メソッドで操作する．URI には，リソースに割り当てられた一意の ID が利用される．そして，リソースの取得には HTTP GET，リソースの作成には HTTP POST，リソースの更新には HTTP PUT が用いられる．他にも，HTTP DELETE や HTTP HEAD などがある．HTTP POST はリソースの生成だけでなく，複雑な処理などにも利用される．リソースを更新する場合，対象のリソースの ID を URI で指定し，ボディ部に更新内容を指定する．

REST では，URI で特定されるリソースに対して，明確に定義されたメソッドを用いることにより，異種のアプリケーションが高い相互接続性で連携することができる．

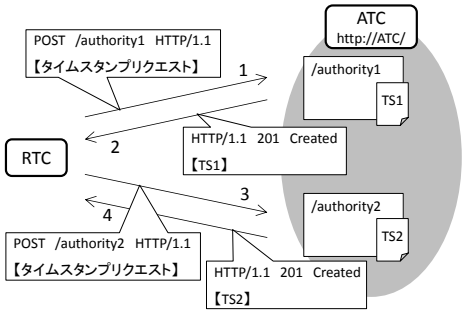

図 7　REST に基づいた時刻認証権限委譲処理

4.2.2. 時刻認証権限委譲処理の設計と実装

REST により，時刻認証権限移譲処理を実装するために，URI と HTTP メソッドの定義を行う．時刻認証権限委譲処理は，ATC の時刻認証処理を 2 回行うことで実現される処理である．本処理の手順は既に[7]で示している．本処理を REST に基づいて実装するために，URI と HTTP メソッドの定義を行う．2 回の時刻認証処理は，処理自体は同じものであるが，状態が異なるため，異なる URI を割り当てる．また，時刻認証処理とは，RTC が ATC に TS を要求し，ATC が TS の生成／発行を行う処理である．つまり，TS の生成を要求している処理となるので，HTTP メソッドは POST を利用する．

図 7 に REST に基づく時刻認証処理を示す．RTC は，http://ATC/authority1/, http://ATC/authoriy2/の順番に HTTP POST でアクセスを行い，時刻認証権限の委譲を受ける．RTC の送信メッセージは，ヘッダ部に HTTP POST，対象のリソースを指定する．そして，ボディ部には TS を要求するデータとして，タイムスタンプリクエスト[2]を ASN.1 エンコーディングしたものをセットする．タイムスタンプリクエストの ASN.1 モジュールは以下のようになる．これは，RFC3161 で規定されている．

```
TimeStampReq ::= SEQUENCE {
   version          INTEGER
   messageImprint   MessageImprint
   reqPolicy        TSAPolicyId
   nonce            INTEGER
   certReq          BOOLEAN
   extensions       [0] IMPLICT Extensions
}
```

ATC は，処理が正常に終了した場合，ヘッダ部にステータス Created(201)をセットし，タイムスタンプレスポンス[2]を返す．このタイムスタンプレスポンス内のタイムスタンプトークンと呼ばれるデータ構造が TS である．タイムスタンプレスポンスの ASN.1 モジュールは以下のようになる．これは，RFC3161 で規定されている．

```
TimeStampResp ::= SEQUENCE {
   status           PKIStatuInfo
   timeStampToken   TimeStampToken
}
```

4.3. TPM による安全なタイムスタンプの生成

RTC は端末内で動作するため，TS 生成時にどの時刻源を使うかが問題となる．もし，端末の内部時計を使うなら，ユーザによる時刻の改ざんが可能となる．[7]では，

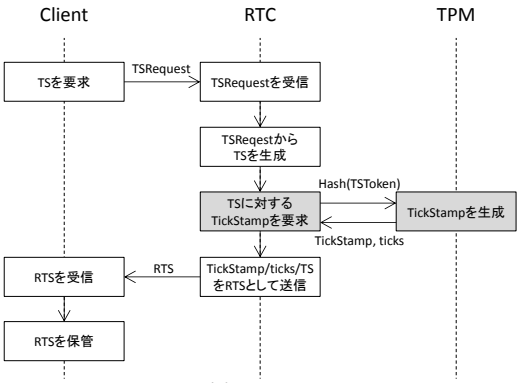

図 8　時刻認証処理フロー

TickCounter 機能を用いた安全な時刻認証の方法について述べた．しかし，実装に関連する部分には触れていなかった．そこで本項では，TickCounter 機能により時刻認証処理を実装する際の問題点を挙げ, その解決策を示す.

4.3.1. TPM（Trusted Platform Module）

TPM[11][12]は，TCG（Trusted Computing Group）が策定した仕様に基づき，端末内部のマザーボードに実装されているセキュリティチップである．セキュリティチップとは, 耐タンパ性を持った IC チップの総称である．つまり，TPM で行われる処理や TPM に格納されているデータは安全性が確保されている.

TPM には，TickCounter 機能と呼ばれる，TPM 起動時からの時間の間隔を保証する機能がある．また，ハッシュ値に対して，TPM が保証する時刻（ticks）と関連付いた署名（TickStamp）を生成する機能を持つ．そこで，提案システムでは，TickStamp を用いて時刻認証を行う.

4.3.2. TickCounter 機能による時刻認証

RTC は Client と同じ端末内で動作するため, 安全に TS の生成を行う必要がある．そこで，TPM の TickCounter 機能を用いる.

図 8 に時刻認証処理フローを示す．RTC は，Client の時刻認証要求に対して TS を生成する．そして，TS を表すデータである TSToken に対して TickStamp を生成する. TickStamp を生成するためのコードを図 9 に示す．RTC は，図 9 の 2 つの関数を呼び出すことによって，TPM から TickStamp を取得する．TPM 内部で取得された ticks により TickStamp を生成しているので，ticks をすり替えて TickStamp を生成することが不可能である．つまり，ticks が表す時刻は安全である.

次に，TS を表すデータである TSToken[2] 内の TSTInfo[2]のフォーマットを表 4 に示す．この TSTInfo に TS の情報が含まれる．IAIK TSP を利用する場合，GenTime フィールドには Date 型の値しか入れることができない．ticks は IAIK 独自の型であるため，Date 型に変換しなければ入れることができない．しかし，TickStamp を検証する場合, ticks が必要となるので, ticks は変換せずに保持しておく必要がある．また，TSToken には署名である TickStamp を入れることができるフィールドも存在しない.

そこで，TickStamp と ticks と TSTInfo を保持するため

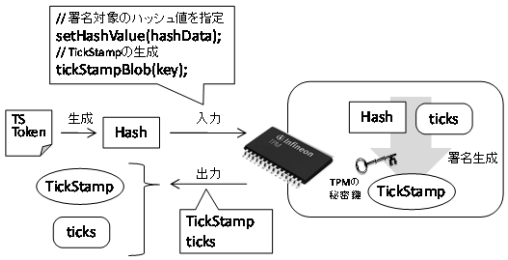

図 9　TickStamp 生成処理

表 4　TSTInfo のフォーマット

Version	TSTokenのバージョン	
Policy	TSAのポリシー	
Message Imprint	ファイルのハッシュ値	
Serial Number	TSTokenに割り振られた一意な番号	
GenTime	保証された時刻	IAIK TSPでは，Date型しか入れることができない
Accuracy	GenTimeからの誤差	
Ordering	GenTimeを用いてTSTokenがソート可能かどうかを示す	
Nonce	ランダムな数字	
Tsa	TSAを識別するための情報	
Extensions	追加情報のための拡張フィールド	IAIK TSPでは，特定の型しか入れることができない

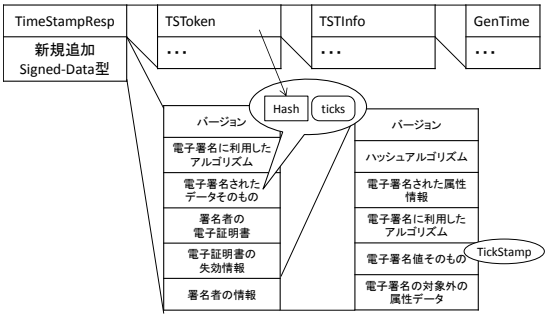

図 10　TPM による時刻認証のためのデータ構造

のデータ構造を定義する. そのデータ構造を図 10 に示す. これを RTS とする．RTS は，TimeStampResp[2] と SignedData[13]をまとめたものとする．SignedData とは，RFC3852 で規定されており，署名を扱うデータ形式である．TickStamp は署名であるので，SignedData 型に格納する.

5. 実装したシステムの安全性

RTC は RTS が改ざんされた場合に検知ができるように，RTS に対する署名である TickStamp を TPM により生成する．図 9 に示すように，TickStamp は TPM 内で生成されているため，RTC は TickStamp 生成処理に関与することはない．つまり，攻撃者は RTC と結託しても TickStamp を不正に生成することができない．また，TPM は耐タンパ性により安全に TickStamp を生成することができるので，攻撃者は TPM 内で実行される

処理に関して不正を行うことができない．以上より，攻撃者はRTSを改ざんしても整合のとれたTickStampを生成することができないため，もしも攻撃者がRTSの改ざんをした場合には検知できる．これにより，表2の色付きの部分の×が○になる．

TPMにより，RTSに対するTickStampを生成することで，表2の色付きの部分が全て○になる．つまり，タイムスタンプサービス提供者のATCが攻撃者と結託する可能性は低く，暗号技術の安全性が低下する前に何らかの対応をとるという状況において，RTCがTPMを用いて時刻認証処理を行うとき，オフライン型タイムスタンプサービスは安全である．

6. おわりに

本稿では，[7]で示した階層型タイムスタンプサービスをもとに，オフライン型タイムスタンプサービスの設計とTSAサービスとクライアントの実装について述べた．まず，オフライン型タイムスタンプサービスの要件を挙げ，それらを実現するためのサービスを構成する主体の機能要件を挙げた．そして，RESTに基づくATCの実装とTPMを用いた安全なTSの生成について述べた．

実装したシステムでは，TPMを用いることにより，攻撃者とRTCの結託／なりすましによるRTSの改ざんを防げることを示し，時刻認証処理が安全に実行できることを示した．

参考文献

[1] 情報処理推進機構,“タイムスタンプ・プロトコルに関する技術調査,”Feb. 2004.

[2] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, ‘‘Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP),’’RFC3161, Aug. 2001.

[3] アマノビジネスソリューションズ株式会社，“アマノタイムスタンプサービス3161，” http://www.e-timing.ne.jp/ (2011/08/16 参照).

[4] PFU，“PFUタイムスタンプサービス，” http://www.pfu.fujitsu.com/tsa/ (2011/08/16 参照).

[5] ドコモエンジニアリング北陸株式会社，“e-DCMタイムスタンプサービス，” http://dcmtsa.nttdocomo.co.jp/ (2011/08/16 参照).

[6] セイコープレシジョン株式会社,“SEIKO Cyber Time,”http://www.seiko-cybertime.jp/ (2011/08/16 参照).

[7] 掛井将平，脇田知彦，毛利公美，白石善明，野口亮司，“階層型タイムスタンプサービスとTPMによる時刻保証方式,”情報処理学会マルチメディア，分散，協調とモバイル（DICOMO2011）シンポジウム，pp. 1002−1015，2011.

[8] SIC, ‘‘TSP / Public Key Infrastructure / Products / Home - Stiftung SIC,’’ (http://jce.iaik.tugraz.at/sic/Products/Public-Key-Infrastructure/TSP, 2011/08/21 参照)

[9] IAIK, ‘‘Trusted Computing for the Java(tm) Platform,’’ (http://trustedjava.sourceforge.net/index.php?item=jtss/readme, 2011/08/21参照)

[10] Bill Burke,“JavaによるRESTfulシステム構築，”オライリー・ジャパン.

[11] 中村智久，東川淳紀，“PC搭載セキュリティチップ(TPM)の概要と最新動向，”IPSJ Magazine, Vol.47, No.5，2006年5月

[12] Trusted Computing Group, ‘‘TPM Main Part1 Design Principles,’’ 2003 年 10 月 2 日.

[13] R. Housley, ‘‘Cryptographic Message Syntax(CMS),’’ RFC3852, July. 2004.