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

掛井将平[†1]　脇田知彦[†2]　毛利公美[†1]
白石善明[†2]　野口亮司[†3]

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．時刻保証要求者は時刻保証が必要になる度に TSA に要求を出さなければならない．一般的に外部に設置されている TSA に処理が集中する従来の二者間モデルでは大量のデータに対する時刻保証は容易ではない．また，端末がオフライン中の時刻保証も困難である．時刻保証を要求した事実を外部に秘匿しておきたい場合もある．

本稿では，まず，TSA の負荷分散を目的としたタイムスタンプサービスの二種類の TSA によるモデルを提案し，その安全性について議論する．そして，そのモデルに基づいた，スケーラブルな時刻保証，端末がオフライン中の時刻保証，時刻保証事実の外部秘匿の 3 点を目的とした端末内での時刻保証を実現する TPM を用いた一方式を提案し，安全性を評価する．

Hierarchical Time Stamp Service and Its Implementation based on TPM

Shohei Kakei[†1]　Tomohiko Wakita[†2]　Masami Mohri[†1]
Yoshiaki Shiraishi[†2]　Ryoji Noguchi[†3]

Time Stamp Service provides verifiability of `proof of existence' and `proof of completeness'. Client requests TSA (Time Stamp Authority) to issue time stamp token as certificate of time guarantee. In two party model of time stamp service, it is not easy to process issuing time stamp token for a large amount of data. Also, it is hard to guarantee the time in offline, and to hide a fact of request to issue a token against outside. This paper proposes a time stamp service model by two kinds of TSA with the aim of load-balancing. We discuss on security of the proposed model. In addition, the paper proposes a method for implementation of the model with the aim of scalable time guarantee, offline time guarantee, and external hiding of time guarantee facts.

†1　岐阜大学
Gifu University
†2　名古屋工業大学
Nagoya Institute of Technology
†3　(株)豊通シスコム
Toyotsu Syscom Corp.

1. はじめに

電子データの作成日時は端末の内部時計から取得される．端末の内部時計は端末利用者により容易に変更できる．また，電子データに付与された時刻は証拠を残すことなく変更することも可能である．つまり，端末が信頼できない状況においては，信頼できる第三者が関与することなく端末で生成された時刻の保証はできない．電子データの時刻保証を行うサービスとして，タイムスタンプサービスがある．タイムスタンプサービスにより，電子データがある特定の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第三者に証明することができる．

タイムスタンプサービスは，外部の信頼できる第三者機関である TSA（Time Stamp Authority）が，信頼できる時刻を保持している TA（Time Authority）から正確な時刻情報を受け取り，電子データに対して信頼できる時刻情報を TST（Time Stamp Token）により付与するサービスである．TSA は外部の機関であることから，時刻保証が必要になる度に TSA へのアクセスが生じる．つまり，時刻保証の要求を行う端末や時刻保証対象の電子データが多くなるほど TSA に時刻保証要求が集中する．また，災害時など通信手段の途絶時には TSA に時刻保証要求を出すことができず，時刻保証を受けることができない．その他にも，TSA に時刻保証対象の電子データのハッシュ値が送付されることに対して，その送付した事実を外部に秘匿しておきたい場合もある．

これらは，外部の機関である TSA が単体で時刻保証を行っていることに原因がある．そこで本稿では，まず，‘TSA の負荷分散’を目的とした二種類の TSA による時刻保証モデルと，そのモデルに基づく‘スケーラブルな時刻保証’，‘オフライン中の時刻保証’，‘時刻保証事実の外部秘匿’を目的とした端末内での時刻保証方式を提案する．なお，端末内での時刻保証には，端末利用者からの不正操作を防止し，安全に時刻の計測ができる仕組みが必要である．そこで，TPM と呼ばれる耐タンパ性を持ったセキュリティチップを利用して実現する．

以下，2 章では，二者間モデルにおける時刻保証について述べ，3 章で階層型時刻保証モデルを提案し，4 章でタイムスタンプの改ざん攻撃に対する安全性について分析する．その結果をもとに 5 章では提案モデルにおける改ざん攻撃が成功しない TPM を用いた方式を提案し，6 章で提案方式の安全性について述べ，最後に 7 章でまとめる．

2. 時刻保証

時刻保証とは，ある電子データにタイムスタンプ（以下，TS とする）を付与することにより，特定の時刻において存在していたことを保証する技術である．また，

その電子データの完全性も保証することができる．TS に求められる要件が文献[1]で次のように示されている．

1. 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．

2. 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．改ざんされていなければ電子データの完全性が示される．

これらの要件を満たす電子署名方式[2][3][4][5][6]やリンキング方式[7]などのタイムスタンプ技術が存在する．

2.1 電子署名に基づく二者間モデル

日本の商用のサービスでは電子署名方式[3][4][5][6][8]が最も多く普及している．電子署名方式は RFC3161 において，PKI TSP として標準化されている．

2.1.1 PKI TSP – RFC3161

RFC3161[2]で，シンプルプロトコルと呼ばれる PKI TSP が標準化されている．PKI TSP は，電子署名に基づくタイムスタンプ・プロトコルである．PKI TSP のモデルを図 1 に示す．PKI TSP は TS 要求者と TSA の二者間モデルである．

2.1.2 TSA(Time Stamp Authority)

TSA とは，外部サーバであり，TS を発行する信頼できる第三者機関である．TSA の内部時計は，世界協定時などの信頼できる時刻源と同期しており，その時刻を時刻保証に利用する．

[TSA の処理内容]

まず，TS 要求者から文書のハッシュ値を受け取る．次に，ハッシュ値と信頼できる時刻源からの時刻から署名を生成する．ハッシュ値，時刻，署名が含まれたタイムスタンプトークンを生成し，これを TS として要求者に送信する．

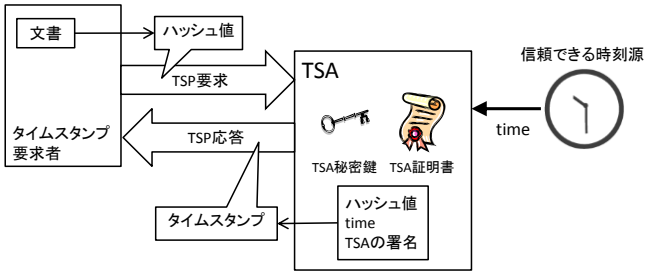

図 1　PKI TSP（文献[1]図 3.1-1 をもとに作成）

[TSA の要件]

RFC3161 では，TSA が満たすべき要件が挙げられている．

1. 信頼に値する時刻の情報源を使うこと
2. 各タイムスタンプトークンに信頼に値する時刻の値を含めること
3. 新たに生成された各タイムスタンプトークンに，一意な数値を含めること
4. 可能であれば，有効なリクエストを要求者から受け取ってからタイムスタンプトークンを作成すること
5. トークンが作成されたときに従ったセキュリティポリシーを指し示すために，各タイムスタンプトークンの中に識別子を含めること
6. データのハッシュ結果についてのみ時刻保証を行うこと
7. 一方向性と衝突困難性をもつハッシュ関数の OID を試験することと，ハッシュ値の長さがハッシュアルゴリズムと整合することを検証すること
8. 上記以外の検証を行わないこと
9. タイムスタンプトークン中に，いかなる要求主体の身元を含まないこと
10. 時刻保証のために生成された鍵を使って各タイムスタンプトークンに署名し，対応する証明書上に示された鍵の属性をもつこと
11. 拡張フィールドを使っている要求者によって依頼された場合，TSA によってサポートされている拡張についてのみ，タイムスタンプトークン中に追加的情報を含めること．これが不可能な場合，TSA は，エラーメッセージで応答すること

2.2 二者間モデルにおける時刻保証

RFC3161 準拠のタイムスタンププロトコル（TSP）による時刻保証では，要求主体は時刻保証対象の電子データのハッシュ値を TSA に送信する．つまり，端末や時刻保証対象の電子データの増加に比例して TSA へのアクセスが増え，TSA の負荷が大きくなる．すなわち，RFC3161 準拠の TSP では端末数やデータ数が増大し，想定していたネットワークの帯域やサーバの処理能力を超える状況になると時刻保証は困難になる．また，TSA は外部サーバであるので，災害時など通信途絶時には TSA に時刻保証要求を出すことができない，つまり，端末がオンラインの状態でしか時刻保証を受けることができない．その他にも，時刻保証の際には TSA に時刻保証対象の電子データのハッシュ値を送信することに対して，その送信した事実を外部に秘匿しておきたい場合がある．これらのことは，時刻保証を TSA のみに依存する二者間モデルに起因している．

そこで，以下では，3 章において‘TSA の負荷分散’を目的とした「階層型時刻保証モデル」を提案し，4 章でその安全性について評価する．5 章において，そのモデルと安全性評価の結果に基づいた‘スケーラブルな時刻保証’，‘オフライン中

の時刻保証’, ‘時刻保証事実の外部秘匿’ を目的とした「端末内での時刻保証方式」の提案を行う．

3. 提案モデル：階層型時刻保証モデル

TSA の負荷分散を目的とした図 2 に示した階層型時刻保証モデルを提案する．

3.1 エンティティ

[**絶対時刻保証者**（ATG）] タイムスタンプサービス提供者が用意するエンティティで，協定世界時などの信頼できる時刻源と同期している．ATG は下位の時刻保証者に対して，時刻保証権限委譲を行うことで時刻保証処理の負荷分散を行う．

[**相対時刻保証者**（RTG）] ATG から委譲された時刻保証権限により，実際に時刻保証処理を行うエンティティ．

[**時刻保証要求者**（Client）] タイムスタンプサービスを受けるエンティティ．RTG に時刻保証要求を出すことによって，時刻保証を受ける．

[**検証者**（Verifier）] 時刻保証権限委譲の正確性や発行された TS の完全性を検証するエンティティ．

3.2 タイムスタンプ（TS）

提案モデルにおける TS を，図 3 に示すような「特定のデータが特定の時刻に存在していたことを証明することを目的として，ATG／RTG が発行する情報で，少なくとも以下の TD，T，ID_{TI}，ID，ICD を含む情報」と定義する．

[TD(Target Data)] 時刻保証対象データ．TS によって時刻の保証を受けるデータである．時刻保証要求データである RD(Request Data)と関連のあるデータであり，RD のハッシュ値や RD そのものである．

[T] 時刻情報. ATS／RTS が TD に対して時刻保証を行ったときの時刻情報である．

[ID_{TI}] 時刻源を表す識別情報．RTG が発行する TS には ATG の識別情報が含まれる．ATG が発行する TS には，ATG が利用する時刻源 TA の識別情報が含まれる．

[ID] ATS／RTS の識別情報.

[ICD(Integrity Check Data)] TS の完全性を確認するためのデータ．TD，T，ID_{TI}，ID からなる電子署名を想定している．

3.3 提案モデルの実現の要件

提案モデルでは，時刻保証権限が委譲された RTG によって時刻保証を行う．提案モデルの TS が信頼できるためには，時刻保証権限が正しく RTG に委譲されていることを確認できればよい．そこで，以下の 3 点を明確にする．

・「いつ」委譲されたのか
・「どのエンティティ」から委譲されたのか
・「どの時刻源」による時刻保証なのか

事実を確認する手段として「いつ」「誰が」という情報は最低限必要である．また，時刻源には，協定世界時や標準日本時などが存在する．そして，あるコミュニティ内だけで信頼できる時刻源などもある．そこで，RTG が委譲された時刻保証権限はどの時刻源によるものなのか明確にしておく．

以上のことを踏まえて，時刻保証権限委譲に関連するエンティティの ATG と RTG

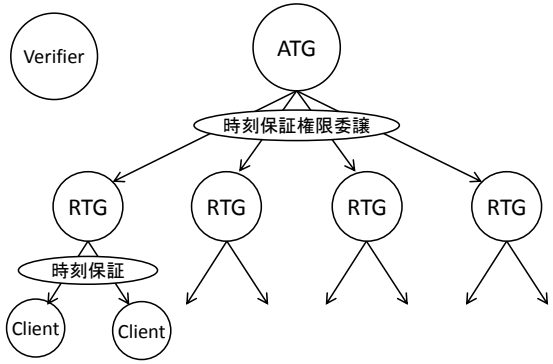

図 2 階層型時刻保証モデル

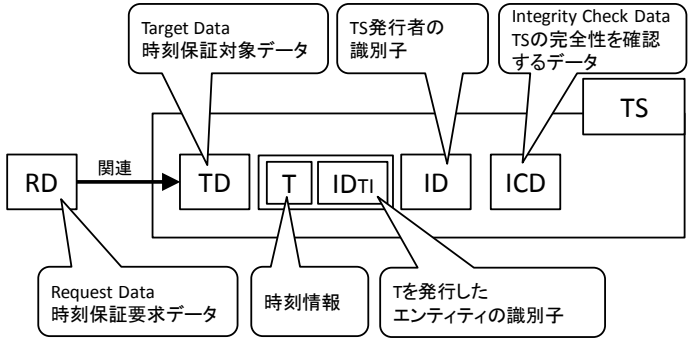

図 3 提案モデルにおける TS

の要件を次のように定義する．

[ATG の要件]

・「委譲日時」「自身の識別情報」「時刻源の識別情報」を含むデータにより，時刻保証権限を委譲できること

[RTG の要件]

・権限を委譲された日時が特定できること

・権限の委譲元が特定できること

・権限の委譲元の時刻源が特定できること

その他，2.1.2 の要件 6, 8 以外の RFC3161 に示されている要件を満たすようにプロトコルを構成する．

3.4 提案モデルの概要

3.4.1 提案モデルの処理の流れ

図 4 に示した提案モデルの処理の流れは次のようになる．

[時刻保証権限委譲処理]

① 時刻保証権限委譲要求

② 内部現在時申請要求

③ 内部現在時申請応答

④ 時刻保証権限委譲応答

RTG は，まず，ATG から時刻保証権限委譲を受けるために①を実行する．次に，ATG は RTG の内部時計の現在の時刻と ATG の内部時計の現在の時刻をリンクさせるために②を実行する．そして，RTG は②に対して，内部時計の現在の時刻を③として ATG に提出する．最後に，ATG と RTG の内部時計がリンクされた情報を④で RTG に送信する．

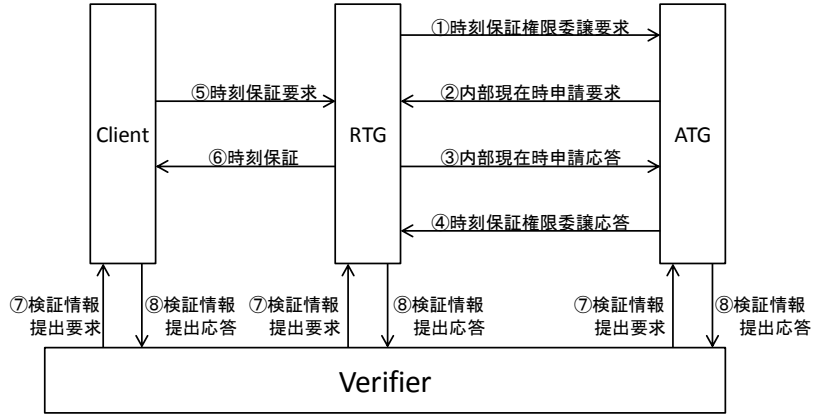

図 4　提案モデルの処理の流れ

[時刻保証処理]

⑤ 時刻保証要求

⑥ 時刻保証

Client は，時刻保証を行いたい時刻保証要求データ RD のハッシュ値 TD を⑤として RTG に送信する．そして，RTG は委譲された時刻保証権限により時刻保証を行う．RTG は受け取ったハッシュ値 TD と RTG の内部時計の時刻情報を関連付ける．この関連付けた情報 TS を⑥として Client に送信する．

[検証処理]

⑦ 検証情報提出要求

⑧ 検証情報提出応答

Verifier は，検証情報を収集するために⑦を実行する．⑦を受けた Client，RTG，ATG は検証情報を⑧として提出する．提出された情報をもとに，Verifier は検証処理を行う．

3.4.2 用語

提案モデルにおいて使用される用語について説明する．

[AT1]　②を実行した時点での ATG の内部時計から取得された時刻情報．

[BRT]　③において提出される RTG の内部時計から取得された時刻情報．

[AT2]　④おいて BRT とリンクされる ATG の内部時計から取得された時刻情報．

[CRT]　⑥において時刻保証に用いられる RTG の内部時計から取得された時刻情報．

[ID_{ATG}]　ATG の識別情報．

[ID_{RTG}]　RTG の識別情報．

[ID_{TA}]　TA（Time Authority）の識別情報．TA とは信頼できる時刻情報を提供するエンティティである．ATG は TA を時刻源として内部時計の調節を行う．また，RTG に対する TA は ATG となる．

[ATS]　ATG が発行する TS．②，④において発行される．図 3 の TS のフォーマットに従う．AT1 を含むものを ATS_{AT1}（図 5），AT2 を含むものを ATS_{AT2}（図 6）と呼ぶ．

[RTS]　RTG が発行する TS．③，⑥において発行される．図 3 の TS のフォーマットに従う．提案モデルにおいて，⑥で発行される RTS が一般的にタイムスタンプと呼ばれるものである．BRT を含むものを RTS_{BRT}（図 7），CRT を含むものを RTS_{CRT}（図 8）と呼ぶ．

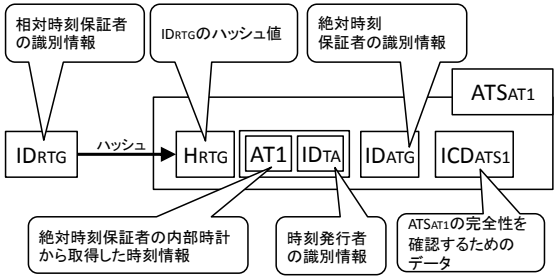

図 5　ATS$_{AT1}$

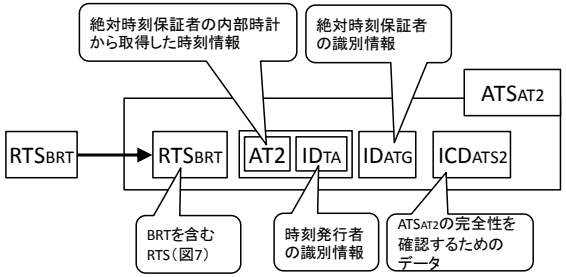

図 6　ATS$_{AT2}$

[ICD$_{ATS}$]　ATS の完全性を確認するためのデータ.

[ICD$_{RTS}$]　RTS の完全性を確認するためのデータ.

3.5　提案モデルの処理

提案モデルは，「時刻保証権限委譲処理」，「時刻保証処理」，「TS の検証処理」の 3 つの処理から構成される．各処理を図 9，図 10，図 11 に示す．ATG／RTG が持つ，TS 発行者の鍵ペアを表 1 に示す.

3.5.1 時刻保証権限委譲処理（図 9）

1. RTG が ID$_{RTG}$ から H$_{RTG}$ を生成
$$H_{RTG} = Hash(ID_{RTG})$$
2. RTG が H$_{RTG}$ を ATG に送信（図 4①に対応）
3. ATG が H$_{RTG}$ を含む ATS$_{AT1}$ を生成
$$ICD_{ATS1} = SIG_{K_{ATG}}(H_{RTG}||AT1||ID_{TA}||ID_{ATG})$$
$$ATS_{AT1} = (H_{RTG}||AT1||ID_{TA}||ID_{ATG}||ICD_{ATS1})$$
4. ATG が ATS$_{AT1}$ を RTG に送信（図 4②に対応）

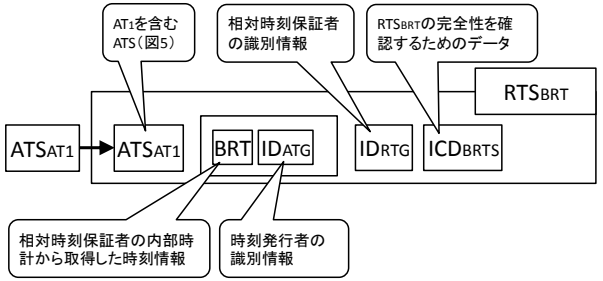

図 7　RTS$_{BRT}$

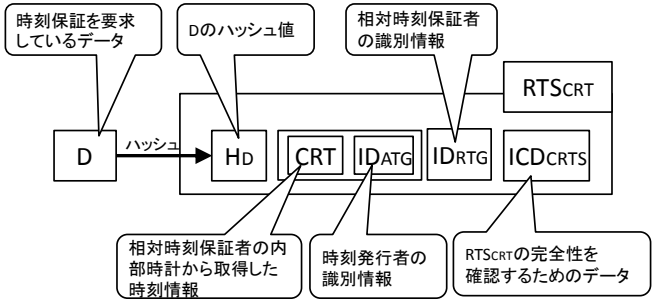

図 8　RTS$_{CRT}$

表 1　TS 発行者の鍵ペア

	秘密鍵	公開鍵
ATG	K_{ATG}	$K_{ATG_{pub}}$
RTG	K_{RTG}	$K_{RTG_{pub}}$

5. RTG が RTS$_{BRT}$ を生成
$$ICD_{BRTS} = SIG_{K_{RTG}}(ATS_{AT1}||BRT||ID_{ATG}||ID_{RTG})$$
$$RTS_{BRT} = (ATS_{AT1}||BRT||ID_{ATG}||ID_{RTG}||ICD_{BRTS})$$
6. RTG が RTS$_{BRT}$ を ATG に送信（図 4③に対応）
7. ATG が RTS$_{BRT}$ を含む ATS$_{AT2}$ を生成

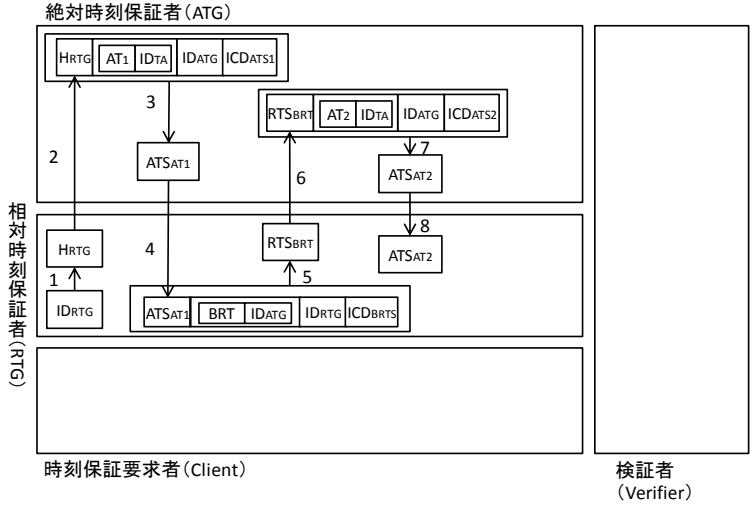

図 9　時刻保証権限委譲処理

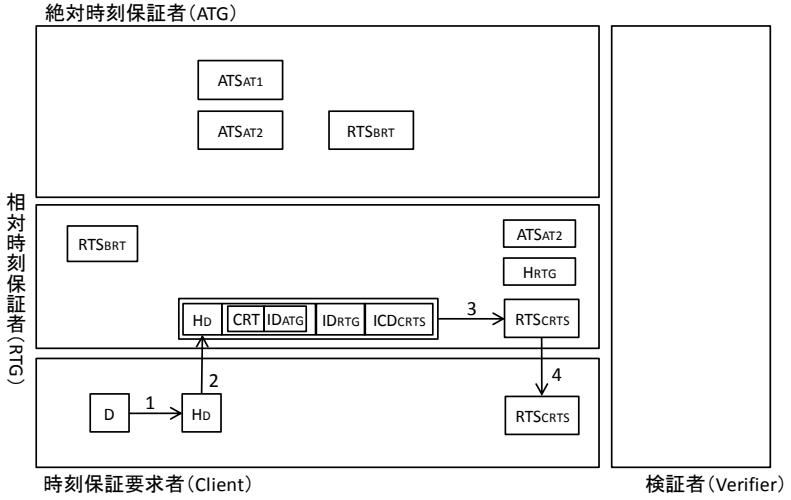

図 10　時刻保証処理

$$\mathrm{ICD_{ATS2}} = SIG_{\mathrm{K_{ATG}}}(\mathrm{RTS_{BRT}}||\mathrm{AT2}||\mathrm{ID_{TA}}||\mathrm{ID_{ATG}})$$

$$\mathrm{ATS_{AT2}} = (\mathrm{RTS_{BRT}}||\mathrm{AT2}||\mathrm{ID_{TA}}||\mathrm{ID_{ATG}}||\mathrm{ICD_{ATS2}})$$

8. ATG が ATSAT2 を RTG に送信（図 4④に対応）

RTG は IDRTG，ATSAT1，RTSBRT，ATSAT2 が検証の際に必要となるので関連付けて保管する．ATG は，ATSAT1，RTSBRT が検証の際に必要となるので関連付けて保管する．

RTG は，BRT と AT2 を関連付けることにより内部時計の調節を行う．ここで，RTG は自身の内部時計を都合のいいように調節し，不正な BRT を利用することが考えられる．提案モデルでは，このような問題が発生しないように，ATG による 2 度の TS（ATSAT1，ATSAT2）の発行を行う．1 つ目のタイムスタンプ ATSAT1 と BRT を関連付けた RTSBRT を生成し，それを ATG に送信することによって，RTSBRT が AT1 から AT2 の間に生成されたことがわかるようにする．

3.5.2 時刻保証処理（図 10）

1. Client が時刻保証要求データ D のハッシュ値 HD を計算

$$\mathrm{H_D} = Hash(\mathrm{D})$$

2. Client が HD を RTG に送信（図 4⑤に対応）
3. RTG が RTSCRT を生成

$$\mathrm{ICD_{CRTS}} = Sig_{\mathrm{K_{RTG}}}(\mathrm{H_D}||\mathrm{CRT}||\mathrm{ID_{ATG}}||\mathrm{ID_{RTG}})$$

$$\mathrm{RTS_{CRT}} = (\mathrm{H_D}||\mathrm{CRT}||\mathrm{ID_{ATG}}||\mathrm{ID_{RTG}}||\mathrm{ICD_{CRTS}})$$

4. RTG が RTSCRT を Client に送信（図 4⑥に対応）

Client は D と RTSCRT が検証の際に必要になるので関連付けて保管する．時刻保証処理は，Client と RTG の二者間モデルである．3.5.1 の ATG から RTG への権限の委譲処理によって，ATG を必要とせずに時刻保証が可能となる．

3.5.3 TS の検証処理（図 11）

1. Verifier が検証必要な情報 ATSAT1，RTSBRT，ATSAT2，RTSCRT，IDRTG，D を各エンティティから収集（図 4⑦⑧に対応）
2. Verifier が RTG から受け取った ATSAT1 の検証（図 12）

[検証処理 A]　ATSAT1 の時刻保証要求データ IDRTG を確認．RTG が持つ IDRTG のハッシュ値と ATSAT1 内の HRTG を比較

$$\mathrm{H_{RTG}} \stackrel{?}{=} Hash(\mathrm{ID_{RTG}})$$

[検証処理 B]　ICDATS1 により，ATSAT1 の完全性を確認

$$Verify_{\mathrm{K_{ATG_{pub}}}}(\mathrm{H_{RTG}}||\mathrm{AT1}||\mathrm{ID_{TA}}||\mathrm{ID_{ATG}}, \mathrm{ICD_{ATS1}})$$

[検証処理 C]　IDATG を持つ ATG に ATSAT1 を送り，ATG は自分が保管しているデータとの整合性を確認（例えば，発行者が保管する TS と比較）

$$\mathrm{ATS_{AT1}} \stackrel{?}{=} \mathrm{ATS_{AT1}}$$

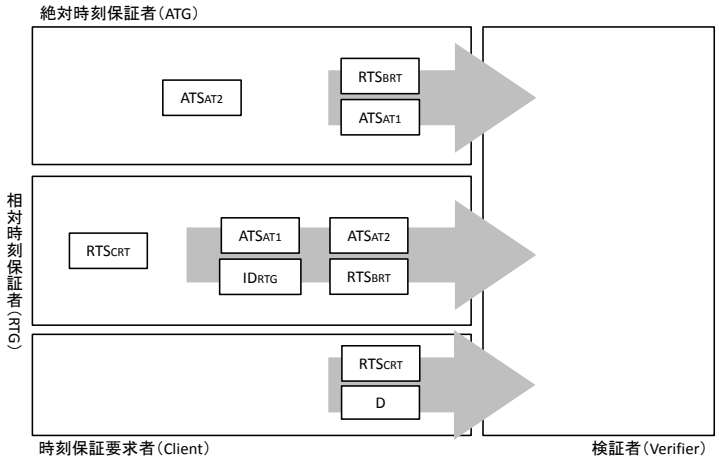

図 11　TS の検証処理

3. Verifier が ATG から受け取った RTS_{BRT} の検証（図 13）
   [検証処理 A]　RTS_{BRT} の時刻保証要求データ ATS_{AT1} を確認. ATG が持つ ATS_{AT1} と RTS_{BRT} 内の ATS_{AT1} を比較

$$ATS_{AT1} \stackrel{?}{=} ATS_{AT1}$$

   [検証処理 B]　ICD_{BRTS} により，RTS_{BRT} の完全性を確認

$$Verify_{K_{RTG_{pub}}}(ATS_{AT1}||BRT||ID_{ATG}||ID_{RTG}, ICD_{BRTS})$$

   [検証処理 C]　ID_{RTG} を持つ RTG に RTS_{BRT} を送り，RTG は自分が保管しているデータとの整合性を確認

$$RTS_{BRT} \stackrel{?}{=} RTS_{BRT}$$

4. Verifier が RTG から受け取った ATS_{AT2} の検証（図 14）
   [検証処理 A]　ATS_{AT2} の時刻保証要求データ RTS_{BRT} を確認. RTG が持つ RTS_{BRT} と ATS_{AT2} 内の RTS_{BRT} を比較

$$RTS_{BRT} \stackrel{?}{=} RTS_{BRT}$$

   [検証処理 B]　ICD_{ATS2} により，ATS_{AT2} の完全性を確認

$$Verify_{K_{ATG_{pub}}}(RTS_{BRT}||AT2||ID_{TA}||ID_{ATG}, ICD_{ATS2})$$

   [検証処理 C]　ID_{ATG} を持つ ATG に ATS_{AT2} を送り，ATG は自分が保管しているデータとの整合性を確認

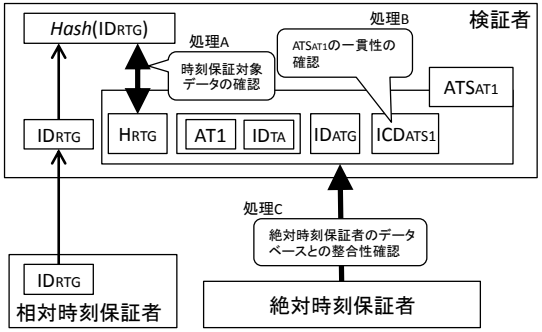

図 12　ATS_{AT1} の検証処理

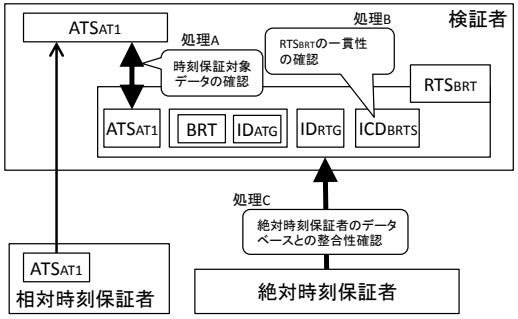

図 13　RTS_{BRT} の検証

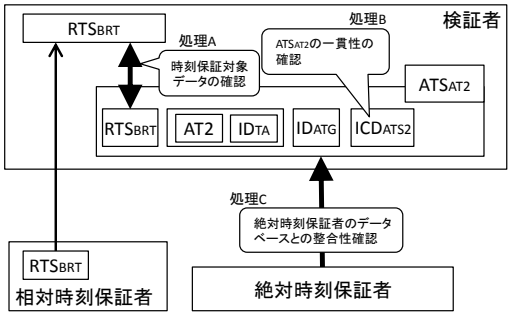

図 14　ATS_{AT2} の検証

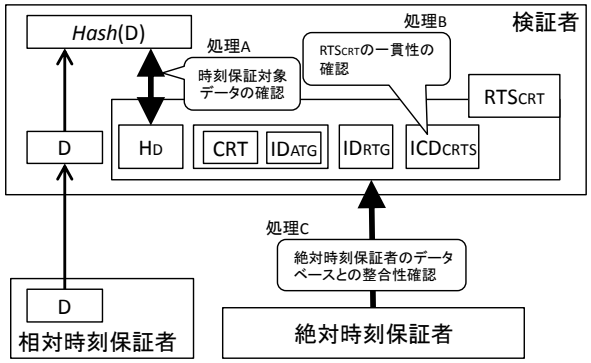

図 15　RTSCRT の検証

$ATS_{AT2} \stackrel{?}{=} ATS_{AT2}$

5. Verifier が Client から受け取った RTSCRT の検証（図 15）

[検証処理 A]　RTSCRT の時刻保証要求データ D を確認

$H_D \stackrel{?}{=} Hash(D)$

[検証処理 B]　ICDCRTS により，RTSCRT の完全性を確認

$$Verify_{K_{RTG_{pub}}}(H_D||CRT||ID_{ATG}||ID_{RTG}, ICD_{CRTS})$$

[検証処理 C]　IDRTG を持つ RTG に RTSCRT を送り，RTG は自分が保管しているデータとの整合性を確認

$RTS_{CRT} \stackrel{?}{=} RTS_{CRT}$

3.6 提案モデルの特徴

3.6.1 TSA の負荷分散

既存の時刻保証の二者間モデルは，想定していたネットワークの帯域やサーバの処理能力を超える状況になると時刻保証が困難になる．提案モデルでは，ATG から時刻保証権限を委譲された RTG が時刻保証を行う．つまり，ATG のみに時刻保証処理の負荷がかからないので，タイムスタンプサービスのスケーラビリティが向上する．

3.6.2 時刻保証権限委譲

複数の RTG がそれぞれ独自の時刻保証権限を持つのではなく，ATG から委譲された時刻保証権限を持つ．ATG が RTG を常に監視しているのではなく，権限の委譲をされた RTG に時刻保証を任せる．

3.6.3 シンプルプロトコル

シンプルプロトコルは，サービス展開しているタイムスタンプ・プロトコル[3][4][5][6]で用いられており，シンプルプロトコルを用いているサービスへの適用，あるいはサービスの拡張が容易となるように，提案モデルでやりとりされる TS はシンプルプロトコルを採用している．

4. 提案モデルの安全性

本章では，提案モデルの安全性について検討する．宇根らは文献[9][10][11]などでタイムスタンプ方式の安全性評価を行っている．本稿では，これらの議論にならって安全性評価を行う．

4.1 攻撃者の設定

[想定する攻撃者]　時刻保証要求者の内の 1 人とする．時刻保証要求者は一般ユーザであり，攻撃者が含まれる可能性が高い．

[攻撃の目的]　検証者によって TS の改ざんを検出されないことを目的とする．TS は検証者によって検証される．その結果から，検証者が正規の TS であるか改ざんされた TS であるかを判定する．TS の改ざんを攻撃者が成功させるためには，検証者に改ざんを検出されてはならない．

[想定する攻撃対象]　攻撃者は「時刻保証対象データ」か「時刻情報」を攻撃対象とする．つまり，時刻保証対象データ TD（HRTG, RTSBRT, ATSAT1, HD）と時刻情報 T（AT1, BRT, AT2, CRT）である．

[想定する攻撃方法]　攻撃対象を改ざんし，改ざん後のデータを含む TS’（ATS’AT1, RTS’BRT，ATS’AT2，RTS’CRT）を生成する．以下の手順で攻撃を行う．

1. 次のどちらか，または両方を実行
   - 時刻保証対象データ TD を別の情報 TD’（H’RTG，RTS’BRT，ATS’AT1，H’D）に改ざん
   - 時刻情報 T を別の情報 T’（AT1’，BRT’，AT2’，CRT’）に改ざん
2. TD’と T’を含む TS’を生成し，TS’と整合性のとれた時刻保証要求データ RD’（ID’RTG，ATS’AT1，RTS’BRT，D’）を生成

[攻撃における前提]　ハッシュ関数は，第 2 原像探索が困難であるとする．IDRTG と ATSAT1，および，D と RTSCRT は，ハッシュ値 HRTG と HD により関連付いている．もし，第 2 原像探索が容易であるとすると，ハッシュ結果が HRTG，HD と同じになる別のデータ ID’RTG，D’が生成可能となる．つまり，時刻保証要求データと時刻保証対象データの関連性がなくなってしまう．また，エンティティ間でやりとりされる通信データは，通信当事者以外に対して守秘性と完全性が確保されている．

[攻撃の条件]　以下の攻撃条件について考える．

- ICD を生成する暗号技術（以下，ICD 生成技術と呼ぶ）の安全性が低下するか，しないか
- 攻撃者が他のエンティティと結託するか，しないか
- 攻撃者が他のエンティティになりすますか，なりすまさないか

ICD 生成技術の安全性が低下するとは，ICD 生成技術に致命的な欠陥が存在し，ICD を生成した者が持つ秘密の情報（署名生成用の鍵など）を知らなくても，ICD を効率的に偽造でき，攻撃者だけがこの方法を知っているという状況である．一方，安全性が低下しないとは，ICD 生成技術に致命的な欠陥が存在しない状態，また，存在しても攻撃者がそれに気づかない状態を意味する．

結託とは，攻撃者と結託対象のエンティティが協力関係にあることを意味する．つまり，結託対象のエンティティは攻撃者の都合のいいように振る舞う．

なりすましとは，ATS_{AT1}，RTS_{BRT}，ATS_{AT2}，RTS_{CRT} を発行した者の識別情報（ID_{ATG}，ID_{RTG}）を攻撃者の識別情報にすり替える行為である．

4.2 提案モデルの安全性の検討

攻撃者の 3 種類の攻撃の条件から，攻撃のパターンは 18 通りある（表 2, 表 3）．18 通りの攻撃パターンに対して安全性を検討する．

検証者は TS（ATS_{AT1}，RTS_{BRT}，ATS_{AT2}，RTS_{CRT}）に対して，検証処理 A，B，C を用いて検証を行う．全ての検証処理において改ざんが検出できなければ，攻撃者の攻撃成功とする．また，1 つでも改ざんを検出した場合，攻撃者の攻撃失敗とする．ここで，ATS_{AT1} と ATS_{AT2}，RTS_{BRT} と RTS_{CRT} はそれぞれ ATG と RTG が発行する同じ構造のデータである．3.5.3 の検証処理による結果はそれぞれ同じになるので，ATS_{AT1} と ATS_{AT2} をまとめて ATS，RTS_{BRT} と RTS_{CRT} をまとめて RTS と呼ぶこととする．また，ATS と RTS をまとめて TS と呼ぶ．

安全性検討の一例として，β-3（表 3）の場合について説明する．攻撃者は，改ざんした TS’（ATS'_{AT1}, RTS'_{BRT}, ATS'_{AT2}, RTS'_{CRT}）と時刻保証要求データ RD’（ID'_{RTG}, ATS'_{AT1}，RTS'_{BRT}，D’）により攻撃を行う．

[検証処理 A]　TS に含まれる TD と RD の関連性を調べる

検証者は，TS’を攻撃者から受け取り，TD’と RD’の関連性を調べる．TD’と RD’は攻撃者が関連付けて生成したデータであるので，検証者はこれらの改ざんを検知できない．つまり，ATS’，RTS’の改ざんを検知できない．

[検証処理 B]　TS に含まれる ICD を用いて，TS の完全性を確認する

ICD は，TS 発行者のみが生成できるデータであるので，攻撃者が TS 発行者と結託／なりすましを行っているかが，検証処理 B の結果に関係する．今回は，β-3 に関して考えるため，攻撃者は ATG と結託しているため，ATG は攻撃者にとって都合のいい ICD’を生成する．つまり，ATS’の改ざんを検知できない．一方，攻撃者が RTG と結託／なりすましを行っていないため，RTG は正しい ICD を生成する．つまり，RTS’の改ざんを検知できる．

[検証処理 C]　TS を TS 発行者に送信し，TS 発行者は自身が持つ TS と送られてきた TS を比較し，その比較結果を検証者に通知

TS 発行者が比較を行い，その結果を検証者に通知するため，攻撃者が TS 発行者と結託／なりすましを行っているかが，検証処理 C の結果に関係する．攻撃者は ATG と結託しているため，ATS’の検証に対して ATG は攻撃者にとって都合にいい結果を検証者に通知する．つまり ATS’の改ざんを検知できない．一方，攻撃者は

表 2　攻撃者と他のエンティティとの結託／なりすましのパターン
（α：ICD 生成技術の安全性が低下する場合）

	攻撃者がRTGと結託	攻撃者がRTGになりすます	攻撃者はRTGと結託／なりすましを行わない
攻撃者がATGと結託	α-1	α-2	α-3
攻撃者がATGになりすます	α-4	α-5	α-6
攻撃者はATGと結託／なりすましを行わない	α-7	α-8	α-9

表 3　攻撃者と他のエンティティとの結託／なりすましのパターン
（β：ICD 生成技術の安全性が低下しない場合）

	攻撃者がRTGと結託	攻撃者がRTGになりすます	攻撃者はRTGと結託／なりすましを行わない
攻撃者がATGと結託	β-1	β-2	β-3
攻撃者がATGになりすます	β-4	β-5	β-6
攻撃者はATGと結託／なりすましを行わない	β-7	β-8	β-9

表 4　β-3 における TS’の改ざん検知の可能性

	検証処理A	検証処理B	検証処理C
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

RTG と結託／なりすましを行っていないため，RTS’の検証に対して RTG は正しい結果を検証者に通知する．つまり，RTS’の改ざんを検知できる．

以上，β-3 における検証処理 A から C より，TS の改ざんを検知できるか，できないかを調べた．その結果を表 4 に示す．このようにして，他の攻撃パターンについても安全性の検討を行う．その結果を表 5 に示す．

ここで，現実の運用を考えて，攻撃者がタイムスタンプサービス提供者である ATG と結託を行う可能性は低く，また，ICD 生成技術の安全性が低下しない，もしくは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する．つまり，表 5 から「攻撃者が ATG と結託しない，かつ，ICD 生成技術の安全性が低下しない」部分に注目する．表 5 の色付きの部分が該当パターンである．この表から，β-4，β-7 において攻撃者は RTS を攻撃することが可能となっている．そこで，β-4，β-7 において，RTS の改ざんを検知でき，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を実現できる方式を提案する．

5. 提案方式：階層型時刻保証モデルの実現

端末内での時刻保証提案方式のモデルを図 16 に示す. ATG である TSA から時刻保証権限が委譲された RTG と Client を同一端末内で動かすことで端末内での時刻保証を行う．このとき，RTG と Client は同じ端末内で動かすため，RTG は端末の内部時計を利用できない．そこで提案方式では TPM を利用する．

5.1 TPM（Trusted Platform Module）

TPM[12][13]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上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 チップの総称であり，解析や改ざんを物理的，論理的に防御する耐性を持つ．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 が持つ基本的な機能として，RSA 演算機能，RSA 鍵生成／格納機能，乱数生成機能やハッシュ演算機能などがある．これらの機能は，TPM1.1b で定義された基本機能である．2005 年には TCG から新たな仕様として TPM1.2 がリリースされた．TPM1.2 は TPM1.1b と互換性がある．新しく追加された機能として，不揮発ストレージ機能，委任機能や TickCounter 機能がある．

5.1.1 TickCounter 機能

TickCounter 機能[13]とは，時間の間隔を計測する機能である．TPM に電力が供給されている間, 一定の間隔でカウンター値が増加する. また, TickCounter 機能では，ある入力に対して，その時点でのカウンター値と関連付いた署名を生成することができる．これは，3.5.1 で述べた RTG による BRT の不正使用防止に必要な機能である. TickCounter 機能を用いることにより, 提案モデルを実現することが可能となる．

表 5　提案モデルの安全性検討結果

パターン		1	2	3	4	5	6	7	8	9
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

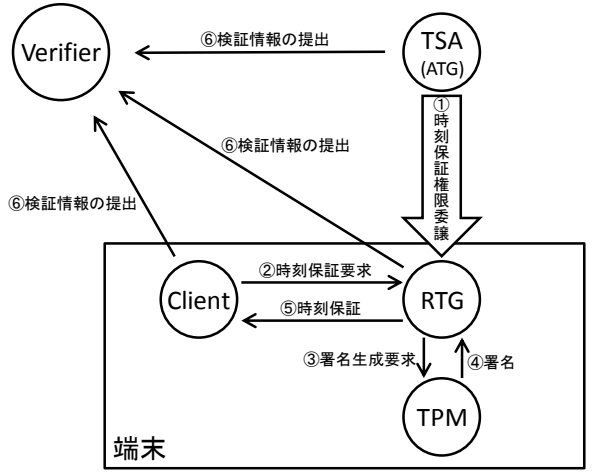

図 16　提案方式のモデル

TickCounter 機能により，以下の情報を取得できる．

[Tick Count Value（TCV）]　Tick セッション内で単調に増加し続けるカウンター値．TCV は 64bit の整数値である．

[Tick Increment Rate（TIR）]　TCV が 1 増加するのにかかる時間，単位はマイクロ秒である．16bit の整数値である．デフォルトでは 1000 で，1 ミリ秒毎に TCV が 1 増加する．

[Tick Session Nonce(TSN)]　Tick セッションを表す値．TSN が同じ値のとき，TCV は同じ時間軸上に存在する．160bit のバイナリデータである．

5.1.2 TPM を用いた RTG による時刻保証

TCV は，単調に増加し続けるカウンター値であるため，TCV から時刻を得ることはできない．そこで，提案方式では，TCV と ATG からの時刻情報をリンクさせることによって，TCV を用いて時刻を算出できるようにする．時刻の算出の過程を図 17 に示した例で説明する．横軸を TCV とし，1 単位時間（t_i-t_{i-1}）を 1 分間とする．

まず，TickCounter 機能を活性化した時点を t_0 とする．次に，BRT を t_{60} とする．このとき，ATG から保証された時刻 AT2 を 15:20 とする．次に，Client のデータに対して時刻保証を行った時刻 CRT を t_{70} とする．このとき，t_{60} から t_{70} へと TCV が 10 増加しているので，10 分経過したことがわかる．また，t_{60} の時点では 15:20 だったので，t_{70} の時点では 15:30 であることがわかる．RTG はこの時刻を時刻保証に利用する．

5.2 提案方式における前提

5.2.1 TPM から得られる時刻情報

RTG が時刻保証に用いる時刻情報は，TickCounter 機能から得られる TCV，TIR，TSN の 3 つからなる．そこで，これらの情報をまとめたものを，提案モデルにおける BRT，CRT とする．

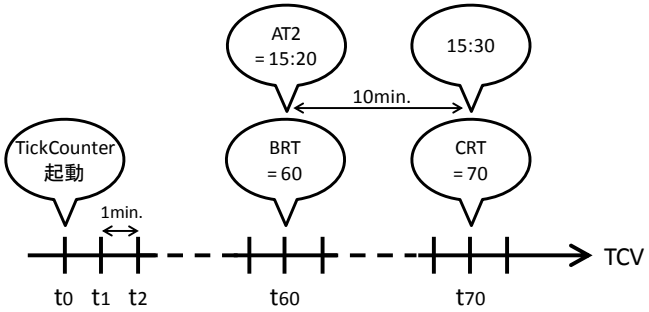

図 17　ATG からの時刻と TCV による現在時刻の算出

表 6　TPM／TSA の鍵ペア

	秘密鍵	公開鍵
TSA	K_{TSA}	$K_{TSA_{pub}}$
TPM	K_{TPM}	$K_{TPM_{pub}}$

5.2.2 TPM／TSA の鍵ペア

提案方式では，TPM が ICDRTS（ICDBRTS，ICDCRTS），TSA が ICDATS（ICDATS1，ICDATS2）を生成する．その際に使用される TPM／TSA の鍵ペアを表 6 に示す．

5.3 提案方式の処理

5.3.1 時刻保証権限委譲処理（図 18）

1. RTG は TPM から TSN を読み込む
2. RTG は TSN と IDRTG のハッシュ値 HTSN，HRTG を計算
   $H_{TSN} = Hash(TSN)$
   $H_{RTG} = Hash(ID_{RTG})$
3. RTG は HTSN，HRTG を TSA に送信
4. TSA は受信したハッシュ値から ATSAT1 を生成／保管．
5. $ICD_{ATS1} = Sig_{K_{TSA}}(H_{TSN}||H_{RTG}||AT1||ID_{TA}||ID_{TSA})$
   $ATS_{AT1} = (H_{TSN}||H_{RTG}||AT1||ID_{TA}||ID_{TSA}||ICD_{ATS1})$
6. ATG は ATSAT1 を RTG に送信
7. RTG は TSA から ATSAT1 を受信／保管
8. RTG は TPM を用いて，ATSAT1 に対して時刻保証
9. TPM は RTG から受け取った ATSAT1，IDTSA，IDRTG と TPM 内部で取得した BRT から ICDBRTS を生成
   $ICD_{BRTS} = Sig_{K_{TPM}}(ATS_{AT1}||BRT||ID_{TSA}||ID_{RTG})$
10. RTG は RTSBRT を生成／保管
    $RTS_{BRT} = (ATS_{AT1}||BRT||ID_{TSA}||ID_{RTG}||ICD_{BRTS})$
11. RTG は TSA に RTSBRT を送信
12. TSA は受信した RTSBRT を検証
    11.1. RTSBRT に含まれている ATSAT1 は自身が発行したものであるか検証
    $Verify_{K_{TSA_{pub}}}(H_{TSN}||H_{RTG}||AT1||ID_{TA}||ID_{TSA}, ICD_{ATS1})$
    11.2. RTSBRT が改ざんされていないか検証
    $Verify_{K_{TPM_{pub}}}(ATS_{AT1}||BRT||ID_{TSA}||ID_{RTG}, ICD_{BRTS})$
    11.3. RTSBRT が ATSAT1 送信後，一定時間 t 以内に送られているか検証
    $AT2 - AT1 \overset{?}{\leqq} t$
13. TSA は検証結果が真なら ATSAT2 を生成／保管
    $ICD_{ATS2} = Sig_{K_{TSA}}(RTS_{BRT}||AT2||ID_{TA}||ID_{TSA})$
    $ATS_{AT2} = (RTS_{BRT}||AT2||ID_{TA}||ID_{TSA}||ICD_{ATS2})$
14. TSA は RTG に ATSAT2 を送信

15. RTG は TSA から受信した ATS_{AT2} を保管

提案方式では，TSN のハッシュ値も TSA に送信している．これは，TSN が一貫して同じ値であることを，端末内で時刻保証するときに確認するためである．TSN が変わった場合は， 5.3.1-処理 1 から行い，時刻保証権限委譲を受けなければならない．

5.3.2 端末内での時刻保証（図 19）

1. Client は時刻保証要求データ D のハッシュ値 H_D を生成

$$H_D = Hash(D)$$

2. Client は H_D を RTG に送信
3. RTG は H_D を Client から受信
4. RTG は H_D，ID_{TSA}，ID_{RTG} を TPM に渡す
5. TPM は RTG から受け取った H_D，ID_{TSA}，ID_{RTG} と TPM 内部で取得した CRT から ICD_{CRTS} を生成

$$ICD_{CRTS} = Sig_{K_{TPM}}(H_D||CRT||ID_{TSA}||ID_{RTG})$$

6. RTG は RTS_{CRT} を生成／保管

$$RTS_{CRT} = (H_D||CRT||ID_{TSA}||ID_{RTG}||ICD_{CRTS})$$

7. RTG は RTS_{CRT} を Client に送信
8. Client は RTG から受信した RTS_{CRT} を保管

5.3.3 検証処理（図 20）

1. Verifier は各エンティティから検証用データを収集
2. Client は Verifier に D，RTS_{CRT} を提出
3. RTG は Verifier に ATS_{AT1}，ATS_{AT2}，TSN，ID_{RTG}，RTS_{BRT} を提出
4. TSA は Verifier に RTS_{BRT}，ATS_{AT1} を提出
5. Verifier は RTG から受け取った ATS_{AT1} を検証

[検証処理 A] 時刻保証対象データと時刻保証要求データに関連があるか調べる

$$H_{TSN} \stackrel{?}{=} Hash(TSN)$$

$$H_{RTG} \stackrel{?}{=} Hash(ID_{RTG})$$

[検証処理 B] ICD_{ATS1} により，ATS_{AT1} の完全性を検証

$$Verify_{K_{TSA_{pub}}}(H_{TSN}||H_{RTG}||AT1||ID_{TA}||ID_{TSA}, ICD_{ATS1})$$

[検証処理 C] TSA に ATS_{AT1} を送信．TSA は自身のデータベースにある ATS_{AT1} と比較し，Verifier に結果を通知

$$ATS_{AT1} \stackrel{?}{=} ATS_{AT1}$$

6. Verifier は TSA から受け取った RTS_{BRT} を検証

[検証処理 A] 時刻保証対象データと時刻保証要求データに関連があるか調べる

$$ATS_{AT1} \stackrel{?}{=} ATS_{AT1}$$

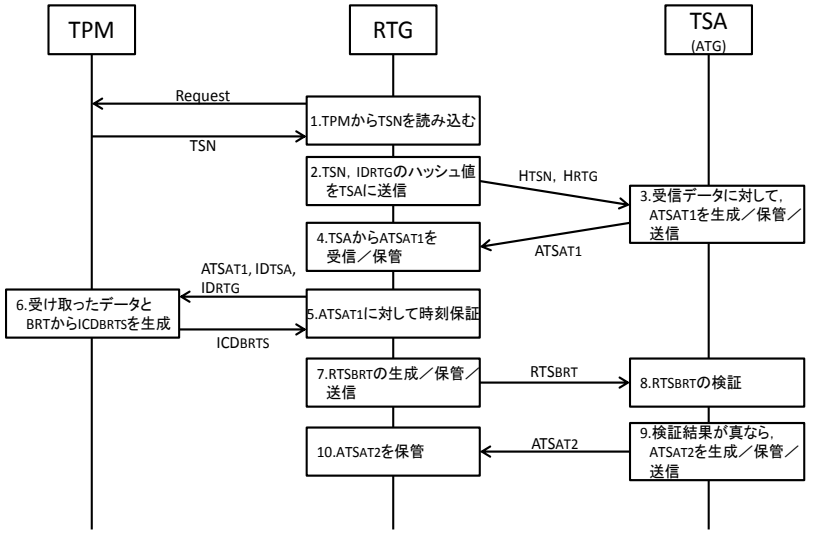

図 18 提案方式における時刻保証権限委譲処理

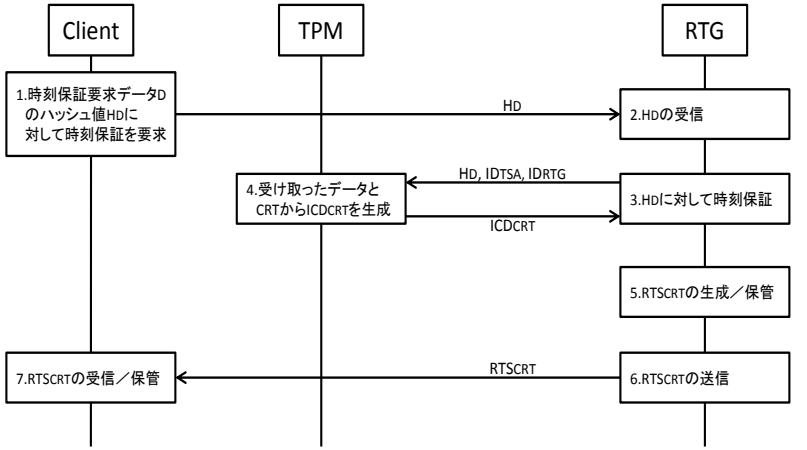

図 19 提案方式における端末内での時刻保証処理

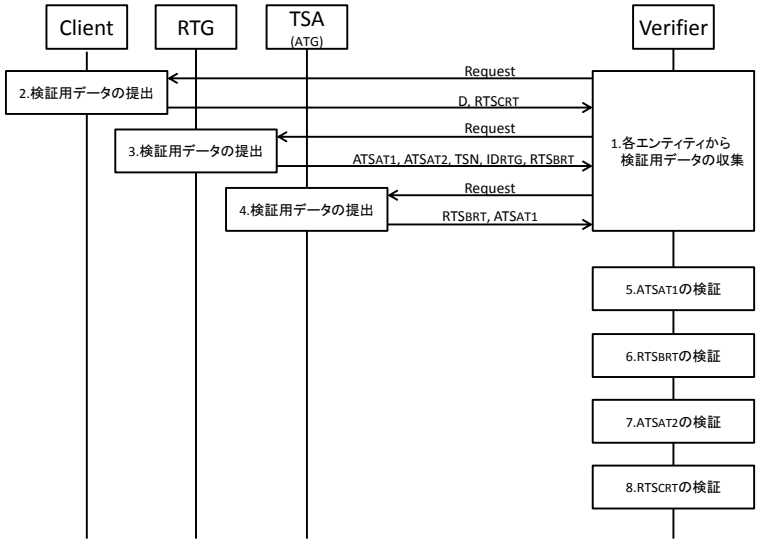

図 20　提案方式における検証処理

[検証処理 B]　ICDBRTS により，RTSBRT の完全性を検証

$$Verify_{\mathrm{K_{TPM_{pub}}}}(\mathrm{ATS_{AT1}||BRT||ID_{TSA}||ID_{RTG}, ICD_{BRTS}})$$

[検証処理 C]　RTG に RTSBRT を送信．RTG は自身のデータベースにある RTSBRT と比較し，Verifier に結果を通知

$$\mathrm{RTS_{BRT}} \stackrel{?}{=} \mathrm{RTS_{BRT}}$$

7. Verifier は RTG から受け取った ATSAT2 を検証

[検証処理 A]　時刻保証対象データと時刻保証要求データに関連があるか調べる

$$\mathrm{RTS_{BRT}} \stackrel{?}{=} \mathrm{RTS_{BRT}}$$

[検証処理 B]　ICDATS2 により，ATSAT2 の完全性を検証

$$Verify_{\mathrm{K_{TSA_{pub}}}}(\mathrm{RTS_{BRT}||AT2||ID_{TA}||ID_{TSA}, ICD_{ATS2}})$$

[検証処理 C]　TSA に ATSAT2 を送信．TSA は自身のデータベースにある ATSAT2 と比較し，Verifier に結果を通知

$$\mathrm{ATS_{AT2}} \stackrel{?}{=} \mathrm{ATS_{AT2}}$$

8. Verifier は Client から受け取った RTSCRT を検証

[検証処理 A]　時刻保証対象データと時刻保証要求データに関連があるか調べる

$$\mathrm{H_D} \stackrel{?}{=} Hash(\mathrm{D})$$

[検証処理 B]　ICDCRTS により，RTSCRT の完全性を検証

$$Verify_{\mathrm{K_{TPM_{pub}}}}(\mathrm{H_D||CRT||ID_{TSA}||ID_{RTG}, ICD_{CRTS}})$$

[検証処理 C]　RTG に RTSCRT を送信．RTG は自身のデータベースにある RTSCRT と比較し，Verifier に結果を通知

$$\mathrm{RTS_{CRT}} \stackrel{?}{=} \mathrm{RTS_{CRT}}$$

5.4　提案方式の実装

提案方式では，端末と TSA の 2 台の PC を利用する．端末の OS は Windows7(64bit)，TSA の OS は CentOS5.4(32bit)である．TPM は端末にのみ搭載されており，バージョン 1.2 である．開発言語は Java を使用し，端末には JRE1.6.0_21，TSA には JRE1.6.0_22 がインストールされている．通信部分は，TCP で実装した．TS 関連の処理は，IAIK が公開している OSS である IAIK-TSP2.1[14]を使用している．TPM 関連の処理は，IAIK が公開している OSS である IAIK jTSS0.6[15]を使用している．

6.　提案方式の安全性

4.2 節において，提案モデルでは，攻撃者が RTG と結託する場合（β-4, β-7）に RTS の改ざんが検知できなかった．そこで，提案方式では，攻撃者が RTG と結託できない，もしくは，結託しても意味がない方式を目指した．

攻撃者が RTG と結託して RTS の改ざんを行うとする．攻撃者は，時刻保証要求データ RD を改ざんした RD’と時刻情報 T を改ざんした T’を，結託している RTG に渡す．RTG はこれらの情報と整合のとれた RTS’を生成するために，TPM を用いて ICD’RTS を生成しようとする．しかし，TPM を用いて ICDRTS を生成する際に，RTG は任意の T を用いることができないため（5.3.2-処理 5），攻撃者は想定していた RTS’を生成することができない．生成される RTS’は，常に正しい時刻情報が含まれた RTS’である．これは，改ざんしたデータに対して正規の手順を踏んで時刻保証を行うことと同等である．つまり，攻撃者は RTG と結託する必要がないということである．また，TPM の持つ署名生成用の鍵は耐タンパ性により守られているため，外部に漏れることはない．

以上より攻撃者は，RTG と結託しても任意の時刻情報による RTS’を生成できず，また，RTG は RTS の署名生成用の鍵を持っていないため，RTG と結託する意味がない．つまり，表 5 の β-4, β-7 において，RTS’の改ざんが検知できるので，提案方式は安全性が確保されている．

7. おわりに

本稿では，まず，‘TSA の負荷分散’を目的とした階層型時刻保証モデルを提案した．そして，提案モデルの安全性をタイムスタンプの改ざん可能性という観点で 18 パターンの攻撃について評価した．その結果，現実的なサービス運用のパターンにおいて, 攻撃者と RTG が結託できなければ階層型タイムスタンプサービスが安全であることを示した．

次に，この提案モデルの安全性評価の結果をもとに，‘スケーラブルな時刻保証’，‘オフライン中の時刻保証’，‘時刻保証事実の外部秘匿’を目的としたタイムスタンプサービスと TPM 搭載端末が連携した端末内での時刻保証方式を提案した. TPM を用いた提案方式では，RTG が不正な RTS を作ることができず，攻撃者は RTG と結託できない方式であることを示した．

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，” 2004 年 2 月.

[2] IPA，“RFC3161，” (http://www.ipa.go.jp/security/rfc/RFC3161JA.html, 2011/05/14 参照)

[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/, 2011/05/15 参照)

[4] PFU，“PFU タイムスタンプサービス，” (http://www.pfu.fujitsu.com/tsa/, 2011/05/15 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/, 2011/05/18 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/, 2011/05/18 参照)

[7] Surety, ``AbsoluteProof Service,'' (http://www.surety.com/, 2011/05/18 参照)

[8] タイムビジネス認定センター，“認定事業者一覧，” (http://www.dekyo.or.jp/tb/list/index.html, 2011/05/17 参照)

[9] 宇根正志，松本勉，“タイムスタンプの安全性と検証手続きとの関連性，” 2001 年暗号と情報セキュリティシンポジウム予稿集，pp. 629-634，2001 年 1 月，

[10] 宇根正志，松本勉，“時刻・証拠付タイムスタンプの安全性と検証情報管理コスト，” 情報処理学会研究報告 CSEC，2001(15)，pp. 31-36，2001 年 2 月.

[11] 宇根正志，松本勉，“タイムスタンプ方式における 10 の安全性クラス，” 情報処理学会研究報告 CSEC，2001(75)，pp. 141-148，2001 年 7 月.

[12] 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.

[13] Trusted Computing Group, ``TPM Main Part1 Design Principles,'' 2003年10月2日.

[14] SIC, ``TSP / Public Key Infrastructure / Products / Home - Stiftung SIC,'' (http://jce.iaik.tugraz.at/sic/Products/Public-Key-Infrastructure/TSP, 2011/05/22 参照)

[15] IAIK, ``Trusted Computing for the Java(tm) Platform,'' (http://trustedjava.sourceforge.net/index.php?item=jtss/readme, 2011/05/22 参照)