1ZA-7

オフライン型タイムスタンプシステムのための Java によるライブラリ

掛井将平†,‡　脇田知彦‡　毛利公美†　白石善明‡　野口亮司††

岐阜大学†　名古屋工業大学‡　（株）豊通シスコム††

1. はじめに

電子データの証拠は‘誰が’，‘何を’，‘いつ’の 3 要素で示される．‘誰が’，‘何を’は電子署名で，‘いつ’は時刻認証で示すことができる[1].

電子データの時刻認証技術として，タイムスタンプ技術がある．タイムスタンプ技術では，時刻認証局（TimeStamp-Authority : TSA）が Client からの要求に応じて時刻認証を行う．RFC3161[2]では，電子署名に基づくタイムスタンプ・プロトコル（Public Key Infrastructure TimeStamp Protocol : PKI TSP）が標準化されている．PKI TSP では，Client が TSA に電子データのハッシュ値を送付し，TSA がそのハッシュ値と信頼できる時刻から署名値を計算することで時刻認証を行う．TSA との通信が必要なので，端末がオフラインのときは時刻認証を受けることができない．適切に時刻認証を受けるには，必要なときに時刻認証を受けられる仕組みが必要である．

我々は，時刻認証の際に TSA との通信が必要ないオフライン型タイムスタンプを提案している[3]．端末内に時刻認証を行うためのエンティティを配置し，そのエンティティが PKI TSP で定義された TSA から時刻認証のための権限の委譲を受けることで時刻認証を行う．ローカルで時刻認証を行うので，端末利用者が不正を行わないようにハードウェアレベルのセキュリティを提供するセキュリティチップ TPM（Trusted Platform Module）[4]を用いる．TPM は外部からの不正な解析に対する耐性，すなわち耐タンパ性を持つ．また，経過時刻を計測できるカウンタ機能を持つ．我々のオフライン型タイムスタンプはこれらの特長を利用して，端末の管理者であっても任意の時刻で時刻認証できないようになっている．

オフライン型タイムスタンプシステムを実装するには，TPM を利用するためのポリシーの設定や TPM で扱うデータの構造などの専門的な知識が必要であり，その実装は容易ではない．

本稿では，オフライン型タイムスタンプシステムのための Java によるライブラリを提案する．本ライブラリにより，Java 開発者はライブラリの使い方を知るだけで本システムを実装できることになる．

2. RFC3161 準拠のタイムスタンプシステム

RFC3161 準拠のタイムスタンプシステムを図 1 に示す．このシステムは Client と TSA の二者間モデルである．TSA は信頼できる第三者機関であり，信頼できる時刻源（協定世界時，日本標準時など）と同期している．時刻認証は次のように行う．Client は，電子データのハッシュ値をタイムスタンプ要求 TSReq として，TSA に送付する．TSA は信頼できる時刻と受け取ったハッシュ値から TSTInfo を生成する．そして，TSTInfo の署名値を計算して，TSToken を生成する．これをタイムスタンプ応答 TSResp として Client に送付する．Client は電子データと TSToken を一緒に保管しておく[1][2]．時刻認証を受けるには，端末はオンラインでなければならない．

このシステムは，TSA は Client からの要求に応じて時刻認証すればよく，複雑な処理を必要としない．このことから，商用のタイムスタンプサービス（[5][6]など）において広く普及している．

企業では，公正・適正な企業活動の実現にはログの収集が重要であり，そのログに対して時刻認証することが求められている[7]．適切な時刻で時刻認証を受けるには，必要な時に時刻認証を受けられる必要がある．また，出先などオフィス以外では必ずしもインターネットに接続できるとは限らず，時刻認証サービスを受けられない場合がある．

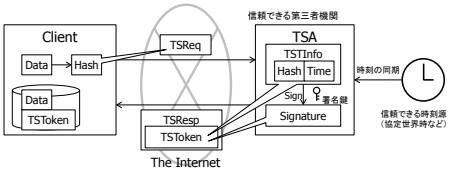

図 1　RFC3161 準拠のタイムスタンプシステム

これらのことから，端末がオフラインのときでも時刻認証するために，ローカルで時刻認証できる仕組みが必要であるが，PKI TSP では想定されていない．そこで，我々は，文献[3]においてローカルで時刻認証するためのオフライン型タイムスタンプを提案している．

3. オフライン型タイムスタンプ

3.1. システム構成

オフライン型タイムスタンプのシステム構成を図 2 に示す．本システムは，5 つのエンティティと 4 つの処理で構成される．

3.1.1.　エンティティ

[TPM 公開鍵管理局]　TPM に対して証明書を発行する Privacy CA，その証明書と端末の紐付けを行う公開鍵管理サーバから構成される．TPM を用いて端末を一意に特定する[8].

[端末認証サーバ]　端末の TPM が TPM 公開鍵管理局に登録されていることを確認する．

[絶対時刻認証者（Absolute Time Certifier : ATC）]　PKI TSP で定義されている TSA．信頼できる第三者機関であり，信頼できる時刻源（協定世界時，日本標準時など）と同期した内部時計を持つ．端末認証サーバで認証された端末に対してのみ時刻認証権限の委譲を行う．

[相対時刻認証者（Relative Time Certifier : RTC）]　ATC から委譲された時刻認証権限をもとに TPM を用いて安全にローカルで時刻認証を行う．TPM で計測された，相対時刻（経過時刻）で時刻認証を行う．

[検証者（Verifier）]　時刻認証権限委譲処理，時刻認証処理が適切に行われたかを，これらの処理で発行された TSToken を検証することで確認する．

3.1.2.　トランザクション

[TPM の登録]　端末管理者が，端末 ID や Privacy CA から発行された AIK 証明書などを TPM 公開鍵管理局に登録する．

[端末認証処理]　端末に TPM が搭載されており，TPM 公開鍵管理局にその TPM が登録されていることを確認する．

[時刻認証権限委譲処理]　ATC が RTC に時刻認証権限の委譲を行う．TPM では相対時刻しか計測できないので，時刻認証処理に必要な絶対時刻を計測できるように，ATC の絶対時

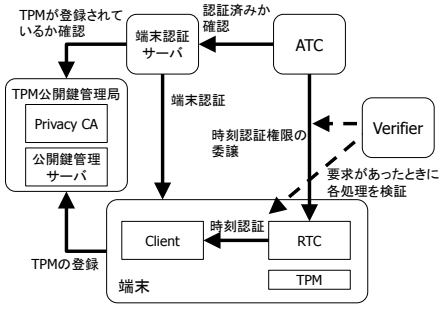

図 2　オフライン型タイムスタンプシステムの構成

The Java Library for Offline TimeStamp System

† Shohei KAKEI and Masami MOHRI・Gifu University

‡ Tomohiko WAKITA and Yoshiaki SHIRAISHI・Nagoya Institute of Technology

†† Ryoji NOGUCHI・Toyotsu Syscom Corp.

刻と RTC の相対時刻の同期を行う．時刻の同期は，ATC と RTC がお互いの時刻情報に対して時刻認証することで行う．RTC は権限の委譲を受ける前に，TPM 公開鍵管理局への端末，TPM の登録と端末認証サーバでの端末認証を受けておく．

[時刻認証処理] Client からの時刻認証要求に対して，RTC が時刻認証を行う．RTC は時刻認証権限委譲処理で同期した相対時刻と絶対時刻を利用する．まず，権限委譲時に同期した相対時刻と現在の相対時刻から経過時刻を求めて，絶対時刻に経過時刻を加えることで現在の絶対時刻を求める．この絶対時刻で時刻認証を行う．

[検証処理] Client の要求に応じて，Verifier が行う．時刻認証権限委譲処理，時刻認証処理が適切に行われたか検証する．Client はこれらの処理で発行された TSToken と時刻認証対象の電子データを Verifier に送付する．Verifier は各 TSToken のハッシュ値を検証することで，一連の処理で生成された TSToken であることを確認し，署名の検証により各 TSToken の偽造・改ざんがないことを確認する．

3.2. システムの実装

オフライン型タイムスタンプシステムの実装を既に行い，TPM を利用して時刻認証できることを確認している．端末の OS は Windows7 で開発言語に Java を用いた．システムの実装には IAIK[9]が公開しているライブラリを用いた．タイムスタンプ関連の処理には IAIK TSP，TPM 関連の処理には IAIK jTSS を用いた．

本システムを実装するには，TPM を扱うためのポリシーの設定，TPM で扱われるデータ構造の理解，用途に応じた鍵の利用が必要であり，これらの知識が必要になる．

本稿では，これらの知識がなくても Java 開発者であれば本システムを実装できるライブラリを提案する．

4. 開発したライブラリ

本稿では，オフライン型タイムスタンプシステムのための Java によるライブラリを提案する．図 3 にオフライン型タイムスタンプシステムの RTC，Client，Verifier の構成を示す．本ライブラリを用いることで網掛け部分を実装することができる．ATC には既存のタイムスタンプサービスを利用する．

4.1. ライブラリの機能

本ライブラリには，TPM 関連，時刻認証関連などの機能が揃っている．これらの機能を利用することで，‘端末認証処理’，‘時刻認証権限委譲処理’，‘時刻認証処理’，‘検証処理’を実装できる．

[TpmUtil クラス] TPM の機能である‘鍵生成’，‘署名値の計算’，‘ハッシュ値の計算’，‘時刻の取得’などを利用するためのクラス．図 4 に，本クラスの利用例として TPM への接続，鍵の生成，ハッシュ値の計算方法を示す．本クラスは，コンストラクタで TPM のオーナーパスワードを指定することで利用できる．端末認証のための TPM に関する情報は，本クラスから取得できる．

[RTCProp クラス／ClientProp クラス] TSToken の要求／生成に必要な情報をプロパティファイルから読み込むためのクラス．プロパティファイルには，ハッシュアルゴリズムや TSA のポリシーID などが以下のようにキーと値のセットとして記述されている．

HASH_ALGORITHM=SHA1
POLICY_ID=1.2.3.4

[TSGenerator クラス] 時刻認証を行うためのクラス．TSTInfo，TSToken，TSResp，TSReq の生成を行う．

[TSVerifier クラス] TSToken の検証を行うクラス．TSToken，時刻認証対象の電子データ，証明書のファイルを指定することで，検証処理を実行する．

[HttpRequester クラス] HTTP で通信するためのクラス．ATC や端末認証サーバと通信するときに利用する．

4.2. ライブラリの利用方法

本ライブラリを利用するために，まずは，BIOS から TPM を利用可能な状態にする．その次に，tpm.msc を実行し，TPM 初期化ウィザードから TPM を有効にし，所有権の設定を行う．3.2.節で示した IAIK が提供する 2 つのライブラリ，jTSS_0.7.zip，iaik_tsp_-eval.zip をダウンロードし，解凍する．次に，Java で TPM を扱うために，解凍した jTSS_0.7 フォルダ内の setup.exe を実行する．最後にダウンロードしたライブラリと本ライブラリにクラスパスを通す．

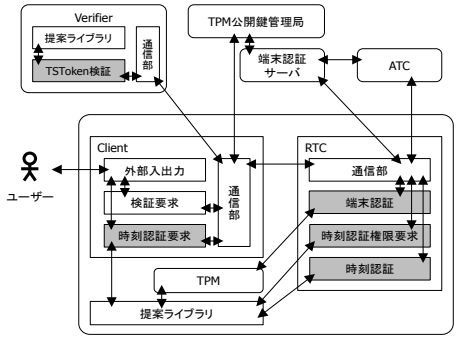

図 3 RTC／Client／Verifer の構成

```
// TPMへの接続
TpmUtil tpmUtil = new TpmUtil("Pass");
// 鍵の生成
TcIRsaKey key = tpmUtil.genTpmKey();
// ハッシュ値の計算
byte[] hash = TpmFunction.calcHash(data, tpmUtil);
```

図 4 TpmUtil クラスの利用例

表 1 ソースコードの行数の比較

	ライブラリ未利用	ライブラリ利用
TPM への接続	18 行	1 行
端末認証処理	27 行	3 行
時刻認証権限委譲処理	192 行	19 行
時刻認証処理	125 行	12 行
検証処理	64 行	10 行

4.3. ライブラリの評価

本ライブラリを利用した場合と利用しない場合のソースコードの行数を表 1 に示す．各処理とも，10 行程度で記述できる．TPM 関連や時刻認証関連の処理をメソッドの呼び出しで実装きるので，ライブラリ利用者は専門的な知識が必要なく，容易にオフライン型タイムスタンプシステムを実装できる．

5. おわりに

本ライブラリを用いることで，TPM の知識が必要なく，オフライン型タイムスタンプシステムを容易に実装できる．ライブラリ利用者は，Java の使い方だけを知っていればよい．

本ライブラリは，TPM の機能を利用するために必要なメソッドが揃っており，他のシステムの実装などにも利用できる．

参考文献

[1] 情報処理推進機構，“タイムスタンプ技術に関する調査報告書，”2004 年 4 月．
[2] C. Adams, P. Cain, D. Pinkas, R. Zuccherato, “Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP),” RFC3161, Aug. 2001.
[3] 掛井将平，脇田知彦，毛利公美，白石善明，野口亮司，“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，”コンピュータセキュリティシンポジウム（CSS2011）論文集，pp. 199-204．2011.
[4] 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．
[5] 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/（2011/12/15参照).
[6] “PFUタイムスタンプサービス，”株式会社PFU, http://www.pfu.fujitsu.com/tsa/（2011/12/15参照).
[7] “Best Practices in Log Management for Security and Compliance,” RSAセキュリティ株式会社，2007.
[8] 脇田知彦，毛利公美，白石善明，野口亮司，“TPMを用いたインベントリ証明書による端末認証のための証明書発行・検証システムの設計と実装，”情報学ワークショップ（WiNF2011）論文集，pp. 155-160. 2011.
[9] IAIK, http://www.iaik.tugraz.at/(2011/12/14参照)