分散時刻認証局グリッドとパラメータ依存性の解析

西川武志[†] 松岡聡[†,††]

デジタル時刻認証はデジタルデータがある時点で『存在していた』,『改竄されていない』ということを証明する手段である．現在メインストリームである集中型タイムスタンプ手法は多数のタイムスタンプ要求が集中することに耐えることができない性能スケーラビリティ上の問題がある．したがって集中型タイムスタンプ手法は分散 DoS 攻撃に弱い．集中型タイムスタンプ手法の性能スケーラビリティ上の問題や分散 DoS 攻撃耐性がないという問題を解決するために分散時刻認証法が提唱されている．しかしながら原子時計のような高価な時刻源を用いることや信頼できる第三者による監査に由来する高コスト性は解決されていない．本論文では我々は $(N, K = L + M, G)$ 手法を用いた TSA Grid と名付けた信頼できる高性能で頑強で安価な分散時刻認証法を提唱する．それは独立の主体によって管理されている peer-to-peer 型の時刻認証プログラムに基づいており，既存の分散時刻認証法のコストの問題を解決する．$(N, K = L + M, G)$ 手法では，N 個の Time Stamping Units（TSU）に G 世代にわたってタイムスタンプ要求が伝搬される．各世代では L 個の信頼できる TSU と M 個のランダムに選んだ TSU からタイムスタンプが要求・応答される．G と L と行ったパラメータの導入により TSU が相互に自律的に監査すること，時刻認証の期待値の推測を可能にしている．また本論文で TSA Grid の基本的なパラメータ依存性について報告する．

Distributed Time-stamping Authority Grid and Analysis of Parameter Dependencies

Takeshi Nishikawa[†] and Satoshi Matsuoka[†,††]

Digital time stamping is a technique to prove the existence of a digital data prior to a specific point in time. The centralized time-stamping scheme which is the main stream at present can not stand up to the concentration of numerous time-stamping requests. So, the centralized time-stamping scheme has vulnerability to the distributed DoS (DDoS) attack. Distributed time stamping schemes have been proposed to solve a performance scalability problem such as tolerance to DDoS attack. They still have high cost problems which are caused by a utilization of atomic clock and by audit of trusted third party. In this paper, we define a reliable, a high-performance, a robust, and inexpensive distributed time stamping scheme. It is named "TSA Grid" with $(N, K = L + M, G)$ scheme and its scheme is based on a network of peer-to-peer time-stamping programs managed by administratively independent entities. It solves the cost problem of proposed distributed time stamping schemes. In $(N, K = L + M, G)$ scheme, one time stamp request propagates for G generation to N Time Stamping Units (TSU). In each generation, L time stamps replies from reliable TSU and M time stamps replies from randomly chosen TSU. The G and the L parameters enabled us to expect authorized time of time-stamping. And they also enabled TSU to audit TSU themselves mutually and automatically. We also investigate basic characterisitic of parameter dependencies of the TSA Grid.

1. はじめに

デジタル時刻認証はデジタルデータがある時点で『存在していた』,『改竄されていない』ということを証明する手段である[1),2)]．その重要性は日増しに増大しており，ビジネス分野では，特に日米では法律で規定された保存電子帳票の正当性証明目的や内部統制での種々の記録の非改竄証明にはデジタル時刻認証の役割が大きなものとなってきている．他方で，デジタル時刻認証は科学技術分野でも知的財産優先権の確保や不正が行われていないことの証明に役立つことが期待されるにもかかわらず，その経済性，高コストであることから普及していない．実験機器の電子化，デジタル化等により出力がデジタルデータとしてのみ保存され，またバイオインフォマティクスのように学問手法自体が，従来の紙媒体の実験ノートに保存しきれな

† 東京工業大学
Tokyo Institute of Technology
†† 国立情報学研究所
National Institute of Informatics

い大量のデータが生成される場面では，元のデジタルデータのハッシュを生成し，元のデータそのものを第三者に渡さずに存在や非改竄の証明ができるデジタル時刻認証は原理的に有用な手段と分かっていながら普及していない．様々なイベントごとに個別にタイムスタンプを発行できれば使い勝手が向上するにもかかわらず，費用の点からまとめざるをえない．

時刻認証局（Time-Stamping Authority: TSA）はデジタルデータのハッシュに対し時刻データを付与し，サーバの署名を施したタイムスタンプを発行し，元データの存在証明，非改竄証明を行う．TSA は，信頼できる時刻源を必要とし，また運用が適切になされているか，時刻を誤ったり，偽証をしたりすることがないよう監査を定期的に受ける必要がある[1),2)]．

現在商用化されている TSA サービス[2)] の多くでは，高信頼時刻の維持と定期的な第三者による監査体制の維持が時刻認証の経済コストを押し上げ利用促進を妨げる 2 大要因となっている．このような経済コストをかけて時刻補正や時刻精度の確認を行っているが，定期的な監査の間隔中での時刻改ざん等の不正の検出は行うことができないという問題がある．

さらに現在実用化されている集中サーバによる時刻認証局では大量のタイムスタンプ要求に応えることは困難である．時刻認証局のタイムスタンプ発行のための計算能力，外部接続ネットワークバンド幅とレイテンシの問題が存在し，多数の要求に応えるにはボトルネックが至るところに存在する．したがって分散 DoS 攻撃に集中サーバ方式が耐性がないことが容易に理解できる．

時刻認証局の分散 DoS 攻撃耐性や性能スケーラビリティの問題を解決する手段として複数の TSA を利用する分散時刻認証法がこれまで提唱されている[5)〜7)]．しかしこれらの手法では多数の TSA を用意しなければならず，設置・運用・監査のコストの問題が存在する．定期的な監査の間隔中での時刻改ざん等の不正に対する対策も考慮されていない．

本論文では，$(N, K = L + M, G)$ と名付けた本分散時刻認証手法を提案，実装し，多数の TSU をクラスタ型計算機の各ノードで動作させた場合の応答性能のパラメータ依存性の解析を行った結果を報告する．本論文で提唱する分散時刻認証局グリッド，TSA Grid は，時刻源としてインターネット上の NTP[8)] サーバを用いた時刻認証ユニット（Time-Stamping Unit: TSU）が，互いに 1 つの時刻認証要求に複数 TSU が複数世代にわたって，NTP により同期した絶対時刻をデジタル署名するタイムスタンプを発行する．全体としてお互いに発行し合ったタイムスタンプに含まれる時刻に基づいて自律的に遅延や時刻の誤りがないかを評価し合い，信用できる TSU のリストを形成してゆくことで分散時刻認証局として監査プロセスをその動作に内在させている．1 つの時刻認証要求に複数のタイムスタンプが発行され，そのタイムスタンプに含まれる時刻に関して多数決や平均値や最頻値を採用することでデジタルデータの時刻認証を行う TSA を構成する．すなわち時刻認証のためのグリッドを時刻認証を利用したい TSU が集まって形成している．

TSA Grid で発行されたタイムスタンプの検証は，分散時刻認証局から源となるタイムスタンプから連環して複数世代にわたって発行された多数のタイムスタンプに記録された時刻から多数決や平均値や最頻値等の統計処理値に基づいている．したがって TSA Grid のタイムスタンプは，タイムスタンプ間の順序を保証するものではない．TSU 数が増えれば増えるほど統計値の分散が低減する．本方式には TSU の総数 N，世代数 G，信頼できる TSU のリストの要素数 L，$(N - L)$ 個の TSU からランダムに選ぶ TSU の数 M という設定パラメータが存在する．各世代ごとのタイムスタンプ発行要求数 K は L と M の和になっている．世代数 G と信頼できる TSU のリストの要素数 L の導入が既存の単一 TSA や分散 TSA に対する優位性をもたらしている．

1 つの時刻認証要求に複数 TSU が G 世代にわたってタイムスタンプを発行することで，既出の分散時刻認証法より少ない TSU の総数 N での運用が可能になりコスト削減が図れる．時間の 1 方向性により複数世代にわたって発行されたタイムスタンプが認証する時刻は後の世代のものは前の世代のものより後の時間となることが要請される．ここに世代が異なるのにまったく同じ時刻や後世代が前世代より先の時刻であればその 2 つの TSU 間でどちらかの時刻に問題があることが検出可能となる．問題があった TSU 以外で多数決をとることで問題のある TSU を特定し修正を行うことが可能になる．

信頼できる L 個の TSU のリスト形成は独立した多数の時刻認証要求の履歴により各 TSU ごとに独立したリストが，第三者によって事前に予測することができないタイミングで形成される．したがって既存の TSA に存在する監査と次の監査の間で時刻改ざん等の不正の可能性を本手法では排除することができる．タイムスタンプ発行と検証を相互に行うこと自体が TSU 相互の監査になる．すなわち on-the-fly で監査をし合うことで，既存 TSA の定期的監査を不要とし，

監査間の時刻詐称可能性を排除し，かつ定期監査のコストを削減できる．既存の分散 TSA では認証される時刻の期待値の推計が困難であるが，本手法では L 個の TSU から得られた統計量から認証される時刻の推定が可能となる．

本論文では，2 章で既存の時刻認証局とその問題点，3 章で TSA Grid の満たすべき要求要件，4 章で $(N, K = L + M, G)$ 分散時刻認証手法の詳細，5 章で評価実験について，それぞれ述べる．

評価実験からは本方式に存在する N，G，L，M という独立パラメータが変化することで認証される時刻がどのように変化するかについて，計算機クラスタ環境上で評価した基本的な結果が明らかとなった．平均応答時間が増大する傾向は，小さな G と N に対して小さい L，M の組合せと，大きな G と N に対して 2 以上の L，M の組合せで観測された．実際の広域ネットワーク上では世代間のネットワーク遅延時間の影響が大きいと推測され，広域ネットワーク上の実証実験では計算機クラスタ環境でさえ応答時間が増大するパラメータの組合せは除外すべきとの予備知識が獲得できた．

2. 既存の時刻認証局とその問題点

2.1 実用化済タイムスタンプ技術

現在，実用化されている時刻認証局は集中サーバに基づくものが多数を占め，耐故障性のためにタイムスタンプサーバがクラスタリングされている時刻認証局も存在するが，ネットワーク上，タイムスタンプ要求を受け付ける URI は特定のものである[3]．したがって，多数のタイムスタンプ要求が一度になされた場合や，DoS 攻撃を受けた場合には，タイムスタンプサーバは要求に応答ができないか，応答できても要求から大きな遅延時間が発生した時刻を持つタイムスタンプ発行となる．上記性能面での問題点以外に標準時との同期や第三者監査に必要な経済コストが高いという問題が存在する．現在商用化されている TSA サービスの多くでは，TSA サーバハードウェアには高精度の時計を内蔵し，その時刻を標準時と専用線で接続した時刻源と同期をとり，かつ第三者の監査を定期的に受けることが時刻認証の経済コストを押し上げる 2 大要因となっている．標準時との同期では日米の標準時はこれまで専用線接続によるサービスを時刻認証局向けに行っており，標準時源までの距離に応じて相応の専用線費用を負担しなければならない．福田らが NTP を用いた追跡可能な時刻配送システム[9] を提唱し，セイコーインスツルも同様な NTP を利用した時刻配信[10] を提供しているが普及は進んでいない．後者は 2006 年に始められたばかりである．第三者による監査では，これまで発行したタイムスタンプどうしが関連しているリンキングプロトコルでは新聞等の公刊物に定期的にタイムスタンプハッシュの一部を公表し，デジタル署名に基づいたシンプルプロトコルでは複数の独立した外部監査人に監査を求めたりしている．

2.2 分散時刻認証局

既存の時刻認証局をネットワーク上に分散させて実現するという提案の多くは，時刻認証局の信頼性の確立の視点に重きを置いたものが多い．

一定数以上の時刻認証局の署名が集めて，多数決，平均値等でタイムスタンプを生成すると定義するものでは，多数が結託あるいは障害が発生しない限り，タイムスタンプを偽造・改竄することが困難であることに基づいている．

これらの分散時刻認証局の仕組みでは各々の時刻認証局が信頼できなくても，多数の時刻認証局が結託する可能性は比較的小さいと考えられるため安全な時刻認証が可能であるというものである．この代表的なものとして秘密鍵分散技術を利用した NTT の分散時刻署名システムがあげられる[4]．これはいかに安全に秘密鍵の部分鍵を複数の TSA に配るかという問題が存在し，かつ複数 TSA の個数は秘密鍵長を部分鍵長で分割する個数となり，非常に少数の TSA から構成され性能スケーラビリティの問題が存在する．これに対し Bonnecaze らが提案する k among n スキーム[5],[6] や Tulone が提唱する RSTS[7] では n 個の時刻認証局から k 個を乱数で選択し多数決により時刻認証を行う仕組みである．この分散時刻認証局に分散 DoS 攻撃を行う者は n サーバすべてに攻撃をかけなければならず，分散 DoS 攻撃に耐性があると唱えている．しかしながら Bonnecaze らが提案する k among n スキームでは認証される時刻の期待値の計算が困難である．Tulone が提唱する RSTS では client から TSA までの round trip time の平均値で期待値が計算できるとしている．しかしこれらの手法では分散時刻認証法で必要となる複数個の TSA の，設置・運用・監査のコストの問題に対する解決策を示していない．また定期的な監査の間隔中での時刻改ざん等の不正に対する対策も考慮されていない．

3. TSA Grid

本論文で提案する TSA Grid では，ネットワーク上に多数の時刻認証要素（Time Stamping Unit: TSU）が相互に時刻認証し合うことで同時に多数の時刻認証

局が結託して時刻を詐称することが困難であること，同じく多数の時刻認証局がネットワーク上に分散していることで分散 DoS 攻撃に耐性を持つという仮定に基づいている．

TSA Grid は，以下の要求，仕様を満たす．

- 各 TSU は RFC3161，“Internet X.509 Public Key Infrastructure Time-Stamp Protocol” [1] に互換の Simple Time-Stamp Unit として動作する．したがって各 TSU が発行するタイムスタンプは一意のシリアル番号を持つこと，
- 各 TSU は TSA Grid 運用規則にのっとって独立に管理されること，
- 各 TSU は独立にインターネット上の NTP サーバと時刻を同期すること，
- 各 TSU は互いに監査し合うこと，
- 各 TSU は特定のセキュリティハードウェアを使用することなく，汎用の計算機上で動作可能であること，
- 各 TSU は絶対時刻をある一定の精度でタイムスタンプとして応答可能であること，
- 簡便，効率的，頑強な統計的処理に基づく検証プロトコルを持つこと，
- Denial of Service（DoS）攻撃に耐性を持つこと，
- 同時多数のリクエストに応えられるようにすること，
- ネットワーク切断・重大遅延等のネットワーク障害に耐性を持つこと，
- タイムスタンプの検証プロセスには十分に長い時間を要しても統計的に十分なサンプルを集めること，
- 発行プロセスは検証プロセスよりプライオリティを高くすること，
- ネットワークトラフィックを過度に消費しないこと，
- 時刻が過失により正確でないノードや悪意を持った第三者による時刻を改ざんしたノードが存在することへの耐性を持つこと，
- TSA Grid 全体として単一障害点を持たないこと．

TSA Grid はタイムスタンプを利用したい主体がタイムスタンプを要求，発行し合うとともに相互に監査を行う TSU を用意することで形成される相互結合ネットワークから構成される．時刻認証を利用したい者が TSU を用意し集まることで TSA Grid を形成する．したがって既存の分散時刻認証法がかかえていた多数の TSU をいかにして用意するかという問題への解決法を提示している．また各 TSU が RFC3161 のインターネット標準に準じることで既存の RFC3161 の時刻認証局ならびにその利用者を TSA Grid へ参加させることができることも目指している．さらに世代数 G を導入することで既存の分散時刻認証法よりも少ない TSU の個数で多数のタイムスタンプ発行が可能となる．

時刻源に関して「インターネットマルチフィード（MFEED）時刻情報提供サービス for Public」[11] 等のようにインターネットを通じて高精度な時刻情報を無償で提供するサービスが存在し，世界協定時 UTC から 1 ミリ秒以内の誤差で時刻を同期することが可能である．各々が相互にタイムスタンプを発行，要求し合うプロセスとその発行されたタイムスタンプに収められた時刻に基づいて各 TSU が他の TSU の信頼性を監査すること，相互監査をシステムに内在することで時刻源および TSA に関する監査費用の低減を図っている．すなわち TSA Grid はタイムスタンプ利用者により自己形成されるシステムである．

4. $(N, K = L + M, G)$ 分散時刻認証手法

ここで，我々が提唱する $(N, K = L + M, G)$ 分散時刻認証手法をタイムスタンプ発行プロセスと検証プロセスについて説明する．

4.1 タイムスタンプ発行プロセス

概略として以下の手順を繰り返す．

クライアントからのタイムスタンプの発行要求からログの記録までの詳細は次のとおりである．

(1) $G = 0$ すなわちクライアントにおいてデジタルデータからハッシュを作成する（mk hash）．
(2) 利用者は RFC3161 に従って時刻認証を行いたいファイルのハッシュからタイムスタンプクエリを作成する（dispatch tsq）．
(3) RFC3161 のタイムスタンプ要求（TSQ）を $G = 1$ である root TSU に発行する（tsq()）．発行を要求した時刻を t_0 とする．
(4) root TSU は，時刻 t_1 を署名したタイムスタンプ応答（TSR）を生成し，a) 非同期に TSR を要求元に返す．このタイムスタンプを root TS と呼ぶ．
(5) 次世代の K 個の TSU は L 個の信頼できるものと $(N - 1 - L)$ 個の TSU からランダムに選んだ M 個のものから選択し TSQ の発行準備をするとともに，拡張プロトコルで保持している現在何世代目かを記録するフラグを 1 減ずる．
(6) 次世代 TSU へ TSQ を要求（g_tsq()）として発行する．

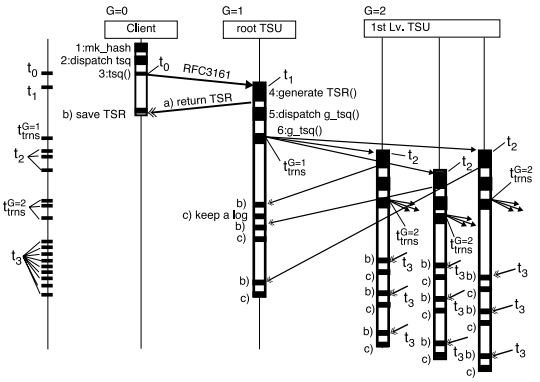

図 1 TSA Grid タイムチャート
Fig. 1 TSA Grid time chart.

(7) 次世代 TSU による TSR 生成，応答が行われ，a) 非同期に TSR を要求元に返す．受け取った前世代 TSU は g_tsq による TSR を手元に保存し，c) どの次世代 TSU からの TSR であったかを記録する．したがってタイムスタンプ時刻は TSQ が TSU に届いて処理される過程で認証されるため，TSQ 要求から TSR 処理，ログの記録までの全体の処理が終わった時刻よりも前の時刻が記録される．
(8) タイムアウトになっていないか，世代数が上限に達していないかを確認し，いずれでもない場合，再び次世代の TSU，K 個をこれまで述べたアルゴリズムに従って選択し TSQ の発行準備をするとともに，現在何世代目かを記録するフラグを 1 減ずる．
(9) 次世代 TSU へ TSQ を $(N, K = L + M, G)$ アルゴリズムによる要求（g_tsq）として発行する．
(10) 以降，(7)〜(9) を G の上限まで繰り返していく．

以上の $G = 2$ までを図示したものを**図 1** に示す．

G が 3 以上のとき，各々の G 世代の TSU は $(G-1)$ 世代から受け取ったタイムスタンプ要求を自己と $(G-1)$ 世代の直接上流の TSU を除いた $(N-2)$ の TSU のうちから $K = (L + M)$ 個を選んでこれまでと同様に世代数を減じて転送要求する．その結果，各 G 世代で $1, K, K^2, \ldots, K^{G-1}$ 個のタイムスタンプ要求がなされ，全体では $(K^G - 1)/(K - 1)$ 個のタイムスタンプ要求がなされる．その結果，最大で $(K^G - 1)/(K - 1)$ 個のタイムスタンプが生成されるが，現実の実装では途中のネットワーク遮断等なんらかの理由により応答がない TSU があることが考えられる．3 世代以上の間では trns TS 要求がループすることがありうる．本 TSA Grid で拡張したプロトコルでは各 TSU はどの TSU からのタイムスタンプ要求を受け応答したか，そのタイムスタンプ要求をどの子世代の TSU に転送したか，それが何世代目だったかを記録する．このようにして各 TSU は親 TSU に trns TS を返答し，子 TSU から trns TS を受け取る．受け取った子 TSU からの最大 K 個のタイムスタンプに署名された時刻を使って信頼できる L 個の TSU のリストを更新する．なお時刻は RFC3161 で規定されたタイムスタンプに署名された時刻の解像度が 1 秒のため，root TS に署名された時刻を t_1 に対する trns TS の時刻との差 Δt は 1 秒単位ごととなる．TSA Grid としての時刻認証は root TS に署名された時刻 t_1 に対し Δt の各種統計処理値（平均値，分散，最頻値，etc.）によって表現された時刻によってなされる．

したがって，本 TSA Grid を使って 1 日の解像度でデジタルデータの時刻認証を行う場合にはタイムスタンプ時刻自体の解像度 1 秒は十分な精度である．

4.2 検証プロセス

TSA Grid において，検証プロセスは，時刻を認証したいデジタルデータのタイムスタンプを検証するプロセス，TSU の時刻が正確かを検証するプロセス，TSU の信頼性を検証するそれぞれが存在する．

4.2.1 タイムスタンプ検証プロセス

TSA Grid においてのタイムスタンプ検証プロセスは，非改竄や存在を証明したいデジタルデータの RFC3161 に基づいたタイムスタンプ要求を新たに生成するか，既存のタイムスタンプ要求を利用して行う．検証したいタイムスタンプをキーとしてタイムスタンプ要求を行ったパスに検証要求を同じく行う．このとき上位の TSU は下位の TSU に検証要求を出し応答を待つ上限時間，タイムアウトをタイムスタンプ発行要求より長くする．タイムスタンプ発行要求は，ただちに上位から下位へ転送されるように実装する．他方で検証プロセスは発行要求よりもプライオリティを低く実装する．これは検証プロセスの負荷が発行プロセスを阻害しないようにし，発行要求を受け付けた TSU が要求時間から遅延が少なくなるよう時刻認証をするためである．

タイムアウト時間内に集まった trns TS の統計値により，root TS の確からしさを検証する．たとえば，Δt の平均値をとった場合は root TS の時刻は $t_1 + EX(\Delta t) \pm VAR(\Delta t)$ と検証される．統計処理は最頻値の算出，四分上位数，最尤度推定等様々なものが目的に応じて利用可能である．

4.2.2 時刻検証プロセス

個々のタイムスタンプ要求の応答結果から Δt が負の値が得られたとする．この場合，要求した自身の時計が遅れているか，要求した相手の時刻が進んでいるかが考えられる．負の Δt を返答した TSU が含まれ

る，Δt が負の値が含まれるパス以外のパスを調べ，該当 TSU が他の上位の（root TS 発行 TSU よりの）trns TS よりも小さな Δt を返しているパスが見つかれば該当 TSU の時刻が進んでいると検出され，そのようなものが見つからなければ自身の時計が遅れている可能性が検出される．前者では該当 TSU に時刻を NTP に同期するよう要求を送り，後者の場合は自己の時計を NTP に同期させる．

4.2.3 信頼性検証プロセス

上記に述べたタイムスタンプの検証プロセスおよび時刻検証プロセスを経て，各 TSU は過去に自分が要求したタイムスタンプ要求に応答した他の TSU の信頼性を評価することができる．定期的に 2 つの検証プロセスを行うことでタイムスタンプ応答の可能性が高い TSU や時刻の誤差が小さい TSU を信頼度が高い TSU として信頼できる TSU リストに加えるだけでなく，リストに保持する TSU の個数 L よりも多くの TSU の信頼性を検証することができる．本スキームでは TSU は独立に管理されており信頼できる TSU のリストは各 TSU が独立に作成し保持するため，悪意ある第三者がある TSU についてこのリストを入手したとしても他の TSU のリストとは異なるため，man-in-middle 攻撃等を回避できる．

4.3 本手法の頑強性

本手法で $G = 1$ とすれば既存の RFC3161 のシンプルプロトコルと同一となるが，多数の TSU が TSA Grid を形成した場合には N 個のなかから 1 つ選ぶため，障害が f 個の TSU で発生しているとすれば f/N となる．$G = 2$ の場合，Δt を評価する際に $N/2$ 未満の個数の TSU が DoS 攻撃にあっても時刻認証が可能であるという頑強性を定義すれば，Bonnecaze らの論文で提案する分散時刻認証局と同じ頑強性となる．すなわち N 個の時刻認証局から K 個のタイムスタンプを得るとするときに，K 個のタイムスタンプから多数決でグローバルタイムスタンプ（GTS）を構成する場合，過半数（$K/2$）を超える時刻認証局からのタイムスタンプが分散 DoS 攻撃でサービス不能もしくは大幅に遅れたタイムスタンプが発行された場合，システムとして時刻証明ができなくなる．その確率は，F をサービス障害の出た時刻認証局の総数，f を集めたタイムスタンプに障害が出た時刻認証局から受け取った個数とすると

$$P\left(f \geq \frac{K}{2}\right) = \frac{1}{\binom{N}{K}} \sum_{\frac{K}{3} \leq i \leq F} \binom{F}{i} \binom{N-F}{K-i} \tag{1}$$

で表される．

G が 3 以上の場合は Bonnecaze らの k among n スキームにおいて，$((N-2)^G - 1)/((N-2)-1) - 1$ の n，最大 $(K^G - 1)/(K-1) - 1$ の k 個のタイムスタンプを得るものと見なすことができる．

4.4 本手法の優位性

本手法の優位性は信頼できる TSU のリスト L を用いることで，最大 $(K^G-1)/(K-1)-1$ のタイムスタンプから得られる統計値の推定が可能なことにある．過去の履歴，または今集めている最大 $(L^G-1)/(L-1)-1$ 個の信頼性が高い TSU からのタイムスタンプによる統計値の推計値が得られる．これにより全体の推計値を推測することが可能になる．k among n スキームでは TSU の中に応答しないものがあれば得られるタイムスタンプによる時刻の期待値は無限大となってしまう．期待値を予測できなくてもサービスを継続するため，Bonnecaze らは障害が発生した TSU の上限を最初の報告では $K/2$ とし，最新の報告では $K/3$ としている．本手法および Bonnecaze ら k among n スキームで利用する統計値として算術平均や多数決の代わりに，幾何平均，最頻度，四分位上位数，四分位上位の最頻度を使って GTS を定義するとより異常値の影響を小さくすることが可能である．

本手法の k among n スキームに対する優位性は k among n スキームが解決策を用意していない，どうやって多数の TSU を用意するか，そしてそれらの維持管理コストを低減するかの解決策を内在していることにある．

5. 評価実験

本論文で提案する $(N, K = L + M, G)$ 分散時刻認証手法を Java および Bouncy Castle API[12] を用いて Servlet として実装した tsagrid プログラムを Tomcat を Servlet/JSP アドオンとして Apache 上で実行させた環境で設計どおりの動作をするか，分散 DoS 攻撃への耐性，および $K = L + M$，G の各パラメータへの動作依存性の評価を行った．

5.1 実験環境

本論文で用いた計算機環境は，東工大松岡研究室の CPU 構成が Athlon2000+ 2way 17 ノード，Opteron 242 2way 19 ノード，Opteron 246 2way 1 ノード，Opteron 242 2way 166 ノード，Opteron 280 2way 19 ノード，Opteron 250 2way 33 ノードの計 256 ノードを複数の 3Com SuperStack 3 Switch 3848 および Dell PowerConnect 5224 Gigabit Ethernet Managed Switch に集線された 1000Base-T ネットワーク

で接続された不均一クラスタ（Prest III）と Orion-Multisystems 社製 Orion DS-96（CPU: Transmeta Efficeon 1.2 GHz，各ノード 2 GB RAM，合計 96 ノード）の均一クラスタである．OS は Linux を採用している．基本動作確認と分散 DoS 攻撃への耐性は前者，$K = L + M$，G の各パラメータへの動作依存性を後者で評価した．

5.2 基本動作検証

基本動作検証では，Prest III クラスタ中 256 ノード中 128 ノードのみで tsagrid を動作させた．残りの 128 ノードでは動作させていないため，応答が返ってこない TSU と判定されるようにした．動作している 128 ノード中 6 ノードに誤った時刻設定がされており，2 ノードは 9 時間進んでおり 3 ノードは 18 時間進んでおり，1 ノードは 27 時間進んでいるようにした．なお評価実験中に時刻は自動的に校正されないようにした．

基本動作の検証として，初めて tsagrid が動作し始める初期状態，各 TSU において近接 TSU リストはノード ID[0-255] 順を近接順序として与えた状態からタイムスタンプ発行要求件数にともなう TSU 応答時間の変化，各 TSU 順序の遷移の様子，誤った時刻が設定された TSU の評価（順位）が下がってゆくかどうかの様子，TSU 応答時間の分布等を調べた．基本パラメータ G，L，M は，世代数 $G = 3$，近接数 $L = 5$，ランダムに選ぶ数 $M = 1$ とした．実験に際しては 128 の TSU のうち 1 つに着目し，タイムスタンプ発行要求を 2,000 回行った．これにより 1 回のタイムスタンプ発行要求ごとに最高 43 のタイムスタンプが得られ，総要求タイムスタンプ数はたかだか 86,000 となり，自己以外の 255 ノード中 1 ノードの順位が最低でもリクエストごとに改訂されることになる．本実験での実装では，応答を返さなかった場合には順位を最下位とするため近接 L に含まれるノードが応答を返さなかった場合，加えて L ノードの順位も更新される．

図 2 に初期状態からタイムスタンプ発行件数にともなう TSU 応答時間の変化の様子を示す．**図 3** に要求元となった TSU における近接 TSU 順位が初期状態からタイムスタンプ発行要求件数ごとにどのように遷移していったかの結果を示す．**図 4** に TSU の応答時間の分布を示す．

5.3 分散 DoS 攻撃への耐性評価

分散 DoS 攻撃への耐性を評価するため前節と同様に $G = 3$，$L = 5$，$M = 1$ として TSA Grid に 2,000 タイムスタンプ発行要求動作をさせ平衡状態とした 128TSU から自己以外の 255TSU にタイムスタンプ

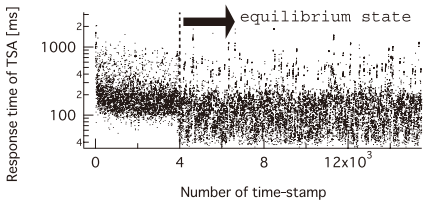

図 2 初期状態からの TSU 応答時間の変化の様子
Fig. 2 TSU response time transition from initial state.

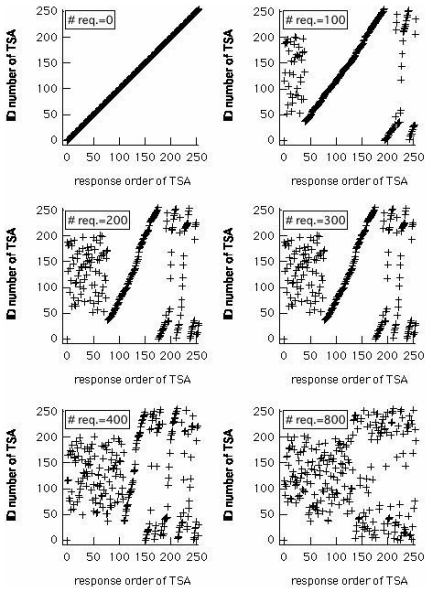

図 3 近接 TSU 順位の発行要求件数ごとの遷移
Fig. 3 Transition status of TSU ranking.

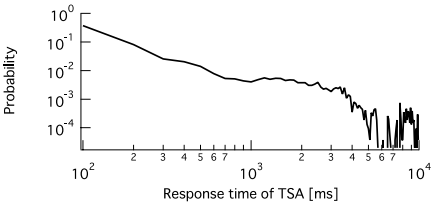

図 4 TSU の応答時間の分布
Fig. 4 Distribution of TSU response time.

要求をバックグラウンドジョブとしていっせいに要求するジョブを各ノード連続 100 件ずつ同時に実行開始して総計 12,800 件，全体で 140,352,000 タイムスタンプリクエストを発行したときの TSA Grid の性能評価を行った．このとき事前に調査し平均して 3 秒の応答遅延が発生するように CPU を消費する演算を 20 プロセス同時実行する負荷を 96 ノードに対して課した．このように通信をクラスタネットワーク内で大量に交差させ，応答すべき 128 ノード中 96 ノードに

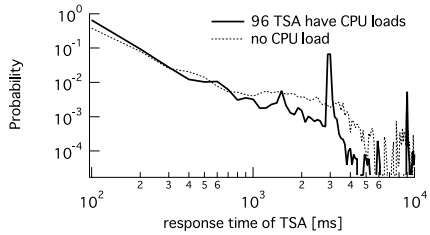

図 5 分散 DoS 攻撃時の TSU の応答時間の分布
Fig. 5 Distribution of TSU response time under distributed DoS attack.

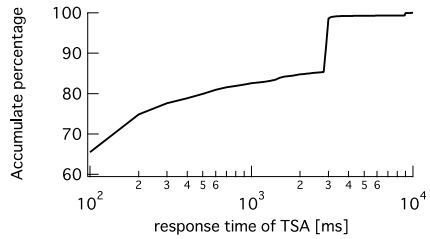

図 6 分散 DoS 攻撃時の TSU の応答時間の積算分布
Fig. 6 Accumulation of TSU response time under distributed DoS attack.

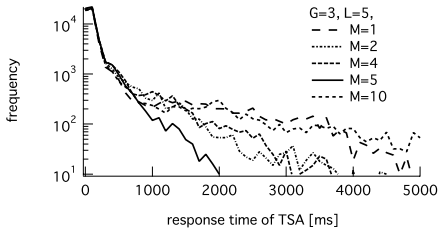

図 7 M への依存性
Fig. 7 Total processing time dependency of generation parameter, M.

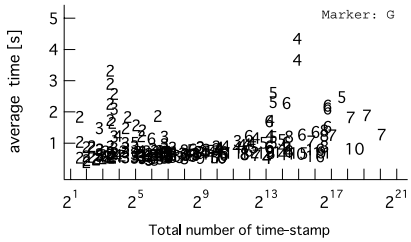

図 8 応答時間の平均値の各パラメータへの依存性
Fig. 8 Average response time dependency of parameters.

応答遅延が発生する負荷を与えることで擬似的に分散 DoS 攻撃が TSA Grid システム全体に行われている状態として実現した．

図 5 に分散 DoS 攻撃時の TSU の応答時間の分布，**図 6** に分散 DoS 攻撃時の TSU の応答時間の積算分布を示す．タイムスタンプ発行総数は 54,064 であり最大総要求タイムスタンプ数 86,000 に対する比率は 62.9%であった．これは Bonnecaze らが提案する分散時刻認証局の k among n スキームを今回と同条件の $n = 256$ ノード中 128 ノードが応答を返さないという状況で $k = 6$ のタイムスタンプが正常に応答する TSU から得られる期待値 1.47%より 43 倍弱も取得確率が高い結果となった．

これは本手法が 2,000 タイムスタンプ発行要求を処理する履歴から信頼できる TSU のリストを動的に形成し障害が発生している 128 ノードにタイムスタンプ発行要求をしなくなるためである．このように履歴から信頼できる TSU のリストを形成することはインターネット上の TSU を使う場合のように大きな遅延やネットワーク切断の確率が LAN を使う計算機クラスタよりもずっと大きな状況でより有効であると推測できる．

5.4 $K = L + M$，G の各パラメータへの動作依存性

DS-96 を用いて様々な G，L，M の組合せについて，応答時間やタイムスタンプに署名された時刻の平均応答時間から TSA Grid の動作の $K = L + M$，G の各パラメータへの動作依存性を検証した．

図 7 に各 TSU の応答時間を $G = 3$，$L = 5$ とし，$M = 1$，2，4，5，10 についてプロットしたものを示す．さらに**図 8** に trns TS に署名された時刻と root TS の差の時間と頻度から計算された平均時間差を，様々な組合せの $K = L + M$，G について，全タイムスタンプの数に対してプロットしたものを示す．なおプロットはそのときの世代 G をマーカーとした．

5.5 考　　察

5.5.1 性 能 評 価

図 2 に着目するとタイムスタンプ発行件数が 4,000 以下では 100 ms 以下の応答時間を示す TSU は少数であるが，4,000 を超えると一気に 100 ms 以下の応答時間を示す TSU が多数を占めるようになった．このとき図 3 に着目すると tsagrid の ID 番号 0 から 35 までと 203 から 255 までのそれぞれの連続範囲で tsagrid が動作していないが，タイムスタンプ要求件数が 100，すなわち要求タイムスタンプ総数が 4,300 までに連続する 2 つ範囲が下位に位置づけられていることに対応すると考えられる．図 3 からは，また，ID 番号が 36 から 202 までに含まれる 39 の tsagrid が動作していないノードや，6 ノードの時刻が進んだ TSU も順位を下げていることが読み取れる．$M = 1$ のため，近接 L に未応答ノードが含まれる初期状態を過ぎると，緩慢に順位評価がなされ，タイムスタンプ要求件数が

800 で動作している 128 の TSU が上位 128 位までにまんべんなく分布するようになった．図 4 から平均の応答時間は 473 ms であり，最頻値は 100 ms である．したがって平均値を用いるより最頻値を用いて GTS を構成する今回の実装がより高速で高精度な時刻認証を実現するといえる．

5.5.2 分散 DoS 攻撃への耐性評価

図 5 に，ある 1 つの TSU での分散 DoS 攻撃への耐性を評価するための負荷をかけた場合の TSU の応答時間の分布（実線）と図 4 で示した無負荷時の TSU の応答時間分布（破線）を示す．1 つの TSU から 255TSU に対して 100 件のタイムスタンプ要求を発行しているが動作しているのは半分の 128TSU なので返答があると期待できるのは 12,800 件，550,400 タイムスタンプリクエストである．結果として該当 TSU に応答があったタイムスタンプリクエストは 65,653 件であり，応答比率は 11.9%となった．クラスタを構成するネットワークのトラフィックバンド幅のピークは 7 MB/s 弱にまで到達した．このとき TSU の応答時間の平均は 647 ms となったが，最頻値は 100 ms を維持した．図 6 に分散 DoS 攻撃シミュレーション時の TSU の応答時間の積算割合を示す．この結果から 1 秒以内の応答時間を満たす取得タイムスタンプの割合は全応答中 83%であり，これは 128 個のアクティブな TSU 中から $K = 6$ 個のタイムスタンプを得るとき，遅延のある 96TSU から 6 個の過半数を得てしまう場合の確率 17%を 100%から引いた値と非常によく一致し，理論どおりの分散 DoS 攻撃に対する耐性を示している．

5.6 各パラメータへの動作依存性解析

図 7 から M が小さくても大きくても TSU の応答時間の分散が大きくなることが分かる．これはタイムスタンプ要求と応答の累計が同じならば M が大きい場合に比べて M が小さい場合は図 2 の平衡状態前のままでいる可能性が高いためと考えられる．M が 10 の場合に 5 より大きくなるのは M が増えた分だけ多数の TSU に対応する処理が増え，処理時間が増大していることが予測され，実際にログを確かめるとそのとおりであった．

図 8 にあるように 2^8 から 2^{12} で平均応答時間が世代数 G によって大きく変動しない領域が見られた．一方でタイムスタンプ要求数が少ない領域，2^1 から 2^6 で，$G = 2$ の場合の平均応答時間が 1 秒を超えるものが多数観測された．これは k among n スキームが k に対して少ない n の場合，k 個の選択した TSU に異常値（ここでは大きな遅延）を含むものがあった場合に平均応答時間が強く影響されていることを傍証するものである．また本提案手法では同じタイムスタンプ要求数なら G が大きい方が，同じ G ならタイムスタンプ要求数が大きい方が，平均応答時間が大きくなることが図 8 から読み取れる．G が大きければ多数の世代を重ねるので TSU 間のネットワークの遅延時間の影響が大きくなることが原因であると考えられる．タイムスタンプ要求数が増えれば 1 つの TSU が処理すべき要求と応答がいずれも増大するため，特に I/O 処理時間がタイムスタンプ応答数に比例して増大するためと考えられ，Tomcat や TSA Grid の実装である tsagrid のログから確認することができた．

なお，当初予想していなかったが，多数のタイムスタンプ要求と応答を繰り返していると Tomcat（Java）のガーベージコレクション（GC）が発生した場合，数秒から数十秒応答が低下することが，$K = L + M$，G の各パラメータへの動作依存性検証中に判明した．しかし，複数の TSU が同時に GC が発生することはまれであり，多数のタイムスタンプ発行においてその影響は軽微であり，かつ GC が発生した TSU は信頼性の高いリスト L から除外されるため，その影響はただちにシステム全体として最小限にくいとめられていることも判明した．

6. ま と め

本論文では，自己に時刻認証局機能を含みネットワーク上の多数の同じプログラムと時刻認証をし合うことで，既存の時刻認証基盤の経済コスト，性能スケーラビリティ，耐障害性の諸問題を解決する $(N, K = L + M, G)$ 分散時刻認証手法を提案した．Java と Bouncy Castle API を用いて実装した tsagrid プログラムが分散時刻認証基盤として基本的な要件を満たしていること，分散 DoS 攻撃に耐性を持つこと，L，M，G の各パラメータに対する動作依存性の解析を行った．本報告で提唱した $(N, K = L + M, G)$ スキームでは，N 個の TSU 中から，G 世代にわたってそのつど，TSU からそれまでの応答履歴から分類された近接 L 個とランダムに選んだ M 個の合計 K 個のノードを選びタイムスタンプを要求するため，既存のまったくランダムに TSU を選ぶ分散 TSA の仕組みでは困難であったタイムスタンプ応答を一定時間内に得ることや，半数の TSU が正常に動作していない場合でも精度が高いタイムスタンプを得られることを示した．今後の課題としては，多数の TSU を実際のインターネット上で運用した場合の実証実験等が考えられる．

謝辞 本研究の一部は NEDO 平成 16 年度産業技術

研究開発助成事業「量子化学グリッド ASP 実証実験」を通じて着想を得て特許出願した発明に基づいて具体的な設計と実装を行ったものである．本報告に先行するプロトタイプの検討と実装に関して議論と AIST スーパクラスタの利用の便宜を図っていただいた，著者の前任所である産業技術総合研究所グリッド研究センターの関口智嗣センター長，横川三津夫前副センター長，工藤知宏クラスタ技術チーム長，田中良夫基盤ソフトチーム長，中田秀基博士の各氏に感謝の意を表する．

参 考 文 献

1) Adams, C., Cain, P., Pinkas, D. and Zuccherato, R.: Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), RFC3161 (2001).
2) タイムビジネス認定センター：時刻認証業務認定事業者一覧．http://www.dekyo.or.jp/tb/
3) Glózik, Z.: Open TSA. http://www.opentsa.org/
4) Takura, A., Ono, S. and Naito, S.: Secure and Trusted Time Stamping Authority, *Proc. IWS'99*, pp.123–128 (1999).
5) Bonnecaze, A., Liardet, P., Gabillon, A. and Blibech, K.: A Distributed Time Stamping Scheme, *Proc. IEEE conference on Signal and Image Technology and Internet Based Systems*, Yaoundé, Cameroon (Nov. 2005).
6) Bonnecaze, A., Liardet, P., Gabillon, A. and Blibech, K.: Secure Time Stamping Schemes: A Distributive Point of View, *Annals of Telecommunications*, Vol.61, No.5–6, pp.662–681 (2006).
7) Tulone, D.: A Scalable and Intrusion-tolerant Digital Time-stamping System, *Proc. 2006 IEEE Int'l Conf. Communications* (*ICC'06*), Vol.5, pp.2357–2363 (June 2006).
8) Mills, D.L.: Network Time Protocol (Version 3), Specification, Implementation and Analysis, RFC1305, (Mar. 1992).
9) 福田晴元，桂木真一郎，石本英隆，小野 諭：NTP を用いた追跡可能な時刻配送システムの設計，情報処理学会デジタルドキュメント研究報告，Vol.2004, No.97, pp.21–27 (2004).
10) セイコーインスツル：SecureNTP. http://www.sii.co.jp/ni/tss/
11) インターネットマルチフィード（MFEED）時刻情報提供サービス for Public. http://www.jst.mfeed.ad.jp/
12) Bouncy Castle Crypto APIs. http://www.bouncycastle.org/

(平成 19 年 1 月 22 日受付)
(平成 19 年 4 月 26 日採録)

西川 武志（正会員）

1992 年慶應義塾大学理工学部計測工学科卒業．1998 年同大学大学院理工学研究科計測工学専攻博士課程修了．博士（工学）．同年 10 月日本学術振興会未来開拓学術研究推進事業リサーチ・アソシエイト（分子科学研究所理論研究系）．2001 年 4 月独立行政法人産業技術総合研究所．2003 年同所グリッド研究センターグリッド応用チーム研究員，2004 年同センター科学技術基盤チーム研究員．2005 年 10 月東京工業大学学術国際情報センター特任助教授，2007 年同特任准教授．分散時刻認証，グリッドアプリケーションサービス，計算機性能評価技術，化学物理の研究開発に従事．電子情報通信学会，分子科学会，日本コンピュータ化学会，日本化学会，日本物理学会，フラーレン・ナノチューブ研究会各会員．

松岡 聡（正会員）

1986 年東京大学理学部情報科学科卒業．1989 年同大学大学院博士課程から，学情報科学科助手に採用，同大学情報工学専攻講師を経て，1996 年東京工業大学情報理工学研究科数理・計算科学専攻助教授．2001 年 4 月東京工業大学学術国際情報センター教授，2002 年国立情報学研究所客員教授併任．博士（理学，東京大学）．高性能システム，並列処理，グリッド，クラスタ計算機，高性能・並列オブジェクト指向言語処理系等の研究に従事．我が国のナショナルグリッドプロジェクトである NAREGI プロジェクトのサブリーダであり，また 2007 年 7 月時点で我が国最高性能のスーパーコンピュータ TSUBAME を構築．1996 年情報処理学会論文賞，1999 年情報処理学会坂井記念賞，2006 年学術振興会賞 (JSPS Award）等を受賞．