L-002

Designated Confirmer Proxy signature を適用した GRID 環境におけるセキュリティ手法の提案・評価

Proposal and evaluation of GRID security environment by designated confirmer proxy signature

猪俣　敦夫† 大山　義人‡ 岡本　健ǂ 岡本　栄司ǂ
Atsuo Inomata Yoshihito Oyama Takeshi Okamoto Eiji Okamoto

1. はじめに

近年、数値計算のニーズやデータベーストランザクションの急激な増加により、従来よりも高速な計算機や大容量ストレージへのアクセスが必要となることが多い。しかし、これらの計算機やストレージは必ずしもローカルに設置されているわけではなく、ネットワークを介して遠隔で利用する必要となることも多々ある。SAN(Storage Area Network)のように、従来よりも高速かつ広帯域ネットワークを利用して遠隔でストレージサービスを提供するための技術も盛んになりつつある。こうした中、大規模な数値計算や大容量データ解析などのリクエストに応じるために、一時的に超高速な計算機を利用した計算環境が必要となることがある。このためには、ローカルに存在している計算機と遠隔に存在する高速な計算機を連携するための技術が必要となる。こうしたニーズに応じる 1 つの解決策として、GRID コンピューティングの研究開発が進められている。GRID の世界では、計算機同士を連携するためのアーキテクチャや、その環境で動作するアプリケーションの研究開発が行われており、既に、Globus Project[1]によって、GRID 環境のミドルウェアの位置づけとされる Globus Toolkit が提供され、いくつかの機能を満たすライブラリ群の整備がされ始めている。

ところで、GRID 環境においては複数の計算機を連携させるためには、各ノードの信頼性を保証する必要がある。これは、例えば悪意のあるノードが追加された場合には、要求した計算結果の妥当性が保証できないことが起きうるためである。Globus Toolkit では、CA(Certificated Authentication)からの証明書をもとに、ノードの信頼性を保証するプロシージャは提供されている。

本論文では、上記セキュリティの視点から、委任署名[2][3]の手法に基づいた GRID 環境に適用するための新たなセキュリティ手法を提案し、その有用性を評価するためにプロトタイプを実装し、実際のネットワーク上で実証実験を実施した。その結果を報告し、最後に今後の GRID 環境におけるセキュリティ機能の可能性について示唆する。

2．GRID 環境でのセキュリティのあり方

GRID 環境において特に重要なことは、リクエストしたジョブに対する結果が信頼できることである。このためには、各ノードやストレージの信頼性が厳格に保証されている必要がある。

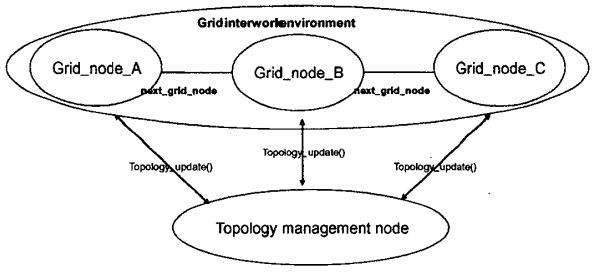
現状、既に CA から発行された証明書を各ノードに配布し、確認されたノードのみが連携できるような仕組みは提供されている。しかし、GRID へのニーズの高まりにより、今後は更なるジョブの多様さ、また動的な計算機の追加・削除が行われることが予想される。このため、動的な GRID ノード構成にも応じたセキュリティ手法について検討することは緊急の課題であると言える。

具体的には、新たなノードを GRID 環境に追加した場合、そのノード自身の信頼性を全てのノードに通知する必要がある。これは、新たにノードが追加されたことにより、GRID 計算環境が変更されたため、既に受け付けられている処理手続きを更新する必要があることが理由である。しかし、現状では一時的に新しい高速なノードを動的に追加した場合には、あらためて CA からの証明書を再度取得して、全ノードへ配布しなければならなく、そのためにリクエストしたジョブをやり直さなくてはならないことも考えられる。我々はその点に着目し、現状の GRID ノードへの証明書配布のプロシージャを拡張し、各ノードの信頼性を保証しながら、動的に GRID ジョブを遂行させるための手法を検討する。

3. GRID 環境におけるセキュリティに対する要件

3.1 Grid topology map

提案手法では、GRID ノードの状態を管理するための GRID topoplogy map を定義する。GRID topology map とは、ネットワークにおけるルーティングテーブルとは異なり、GRID 環境において、各ノードの論理的な関係をマッピングしたテーブルである。図 1 に、Grid topology map の一例を示す。上部がマップを概念的に示した図であり、下部はノード間の関係を示したテーブルを示す。このテーブルを Topology map management ノードによって管理される。



	Grid_node_A	Grid_node_B	Grid_node_C
Grid_node_A		Next_node	Via_node_B
Grid_node_B	Next_node		Next_node
Grid_node_C	Via_node_B	Next_node	

図 1　Grid topology map

†, 独立行政法人 科学技術振興機構, Japan Science and Technology agency
‡, 北海道大学,Hokkaido University
ǂ, 筑波大学,University of Tsukuba

Topology map management ノードとは、トポロジマップを管理するノードであり、各 GRID ノードから指定された時間ごとにこのテーブルが参照される。マップを参照した各ノードは、指定された一定時間の間キャッシュする。図 1 は、Node_A と Node_B は論理的に"next-GRID_node"であることを示し、Node_A と Node_C は、ネットワークを介して接続はされているものの、GRID 環境上においては"next-GRID_node"ではないことを示す。すなわち、ルーティングにおける"nexthop"とは異なることに注意する。

4. Designated confirmer proxy signature に基づいた GRID signature 手法の提案、および設計

4.1 Designated confirmer signature とは

従来の公開鍵署名方式では、署名者が管理する秘密鍵と公開鍵のペアリングをもとに、実際にやり取りが行われたメッセージの正当性を確認するための手段が提供されている。しかし、やり取りされるメッセージが増大するにつれて、署名者の負担が大きくなることが懸念される。そこで、署名者が代理人(Confirmer)を選択し、Confirmer が代理で証明できるような署名を作成して、検証者に渡し検証が行われる仕組みをとる Designated confirmer signature が提案されている[4][5]。提案手法では、Confirmer と検証者の間にやり取りされるプロトコル中に、検証手続きが含まれる。

4.2 委任署名(Proxy signature)とは

メッセージの正当性検証の負荷の増大にあわせて、さらに署名作成の手間も増大する。公開鍵署名基盤において、重要なことはオリジナルの署名者の秘密鍵を他人に知られてはならない。そこで、オリジナルの署名者の秘密鍵から代理署名用の秘密鍵と付属情報を算出し、複数の代理署名者に渡す。そして代理署名者が署名を作成し、それが検証者に渡されメッセージの検証が行われる仕組みをとる委任署名も提案されている[2][3]。我々が提案する GRID 環境においては、オリジナルの署名者が GRID 計算のリクエストを出したノードとなり、その代理署名者が各 GRID ノードとなる。すなわち GRID ノード全てにおいて、新たな署名作成作業が同時に行えるため、セキュリティを考慮した上で動的な GRID ジョブのリクエストに比較的容易に応じることが可能となる。

4.3 Designated confirmer proxy signature の適用

そこで、複数の代理人により署名作成作業を可能とした手法として、委任署名の手法を Designated confirmer 署名に埋め込んだ Designated confirmer proxy signature 手法を利用した新たな GRID 証明書配布手続き手法を検討した。その詳細を図 2 に示す。

オリジナルの署名者は、リクエストを出す Grid_requested_node であり、元々の秘密鍵と公開鍵のペアを有する。続いて、GRID 計算の初期の時点で証明書を配布する Globus-proxy-init()メソッドを我々が拡張した Globus-designated-confirmer-proxy-init()を実行することにより、Designated confirmer proxy signer にて署名作成の準備が行われる。この Designated confirmer proxy signer とは、各 GRID ノードのことである。各 GRID ノードでは、署名作成を行う。この手段には、我々の実装した message(secret key, optional);メソッドを利用する。

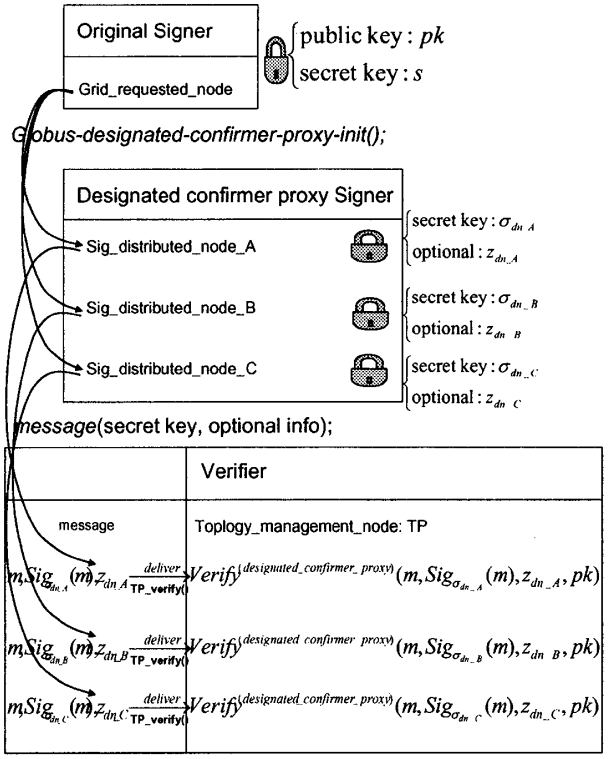


図 2　提案する GRID 環境における署名作成手法

署名が作成されると、実際に検証者に渡され検証が行われる。なお、検証者は Topology management node となる。検証者へは、トポロジマップ情報の更新プロトコルである TP_update()メソッドにより渡され、検証作業は TP_verify()メソッドにより行われる。この手続きにより各ノードの信頼性が確認される。

ところでトポロジマップとは、各 GRID ノードの関係を示したマップであり、GRID 上での"next-GRID_node"が示されている。既に、ジョブが進められている状況において、新たな GRID 計算ノードが追加された状況に応じるために、さらに Parent_node という概念を導入する。Parent_node とは、GRID 環境に他ノードと連携する際に、自ノード情報が通知されるノードを指し、あらかじめ新たに追加されるノードが Parent_node を指定する。通知を行った後、そのノードは指定した Parent_node からあらかじめ配布済の証明書の配布を受けることで、Parent_node との間で検証が行われる。もちろん、この手続きも Designated confirmer proxy signature 手法をもとに手続きが進められるため、そのノードの妥当性を保証することが可能となる。

5. プロトタイプの実装、および実証実験

プロトタイプを実装するにあたり、実装を容易にする観点から、証明書は第 3 者となる機関から発行したものを再利用する形態にした。プロトタイプは、Linux RedHat 上にて C を用いて実装した。また、Globus Toolkit 3.0.2 をベースとして、証明書配布手続きとノードが追加された場合の処理部分を導入するために、一部メソッドを拡張した。

また、Globus のサービス（ジョブ管理やセキュリティ）を用いて、複数ノードを結合し MPI アプリケーションを実行させるために、MPICH-2 が動作する環境を準備した。プロトタイプの詳細については、以下の節に述べる。

5.1 プロトタイプ

実装したプロトタイプの概要を図 3 に示す。本プロトタイプでは、Topology management node（以降 TP）と各 GRID ノードと CA の 3 つのノードで構成される。CA は、第 3 者機関より発行された証明書を格納し、TP に応じて配布が行われる。さらに、TP では、前述した Topology map を管理している。なお、各 GRID ノード間の関係を示す Topology テーブルは topology map 中に含まれる。各 GRID ノードは指定された時間ごとに、TP に対して Topology map をクエリーし、その状態を更新する。この手段には、Topology_update()メソッドが呼び出される。

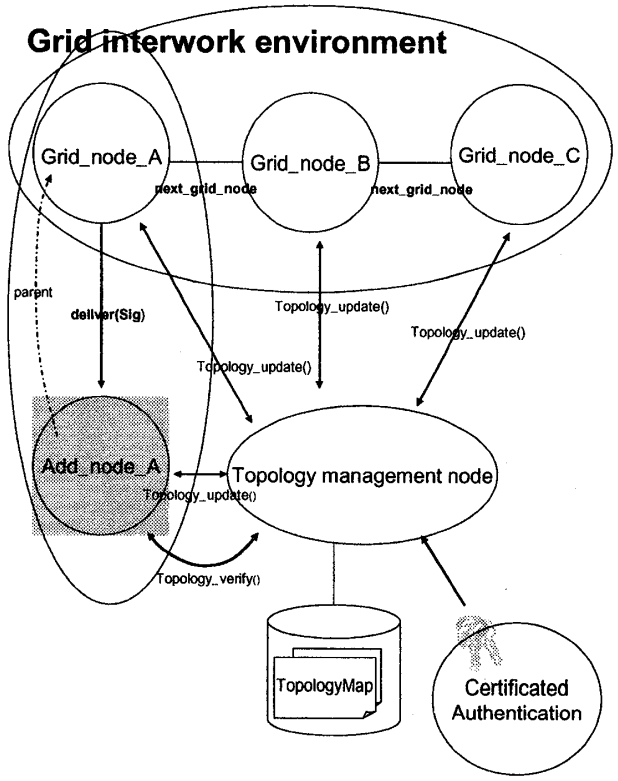


図 3　プロトタイプシステムの概要およびフロー

既に GRID ジョブが進められている中で、新しいノードが追加された場合の手続きを以下に述べる。

1．新しく追加されるノードの設定ファイル中に、どちらの GRID ノードを Parent_node とするかを指定する。ネットワークに接続されると、まずは、自ノードの情報（IP アドレスおよび Parent_node を含む付属情報）を Topology management node に通知する。図 3 の例では、Parent_node として、Grid_node_A が指定されている。
2．Topology management node は、トポロジマップ情報を更新する。
3．各 GRID ノードは、指定された時間ごとにトポロジマップ情報を更新する。もし、トポロジマップが更新されていた場合、ローカルにキャッシュしている情報を更新する（本プロトタイプでは、更新されていたかどうかはファイルの差分から判断している）
4．各 GRID ノードは更新されたトポロジマップを参照し、もし Parent_node として自ノードが指定されていた場合には、既に CA より配布されている証明書をその新たに追加されたノードに渡す。
5．最後に、その証明書を検証者となる Topology management node との間で検証作業が行われる。検証作業が完了すると、GRID 計算ジョブのリクエストを出したユーザに通知を行う（今回の実験では、Original Signer が Grid ジョブリクエストを出したノードに通知する）

5.2 実証実験

実装したプロトタイプをもとに、提案した手法の有効性を検証するために実証実験を実施した。実証実験では、実際のネットワーク上への適用を考慮した上で、実験系を構築した。詳細を以下に述べる。

5.2.1 実験系

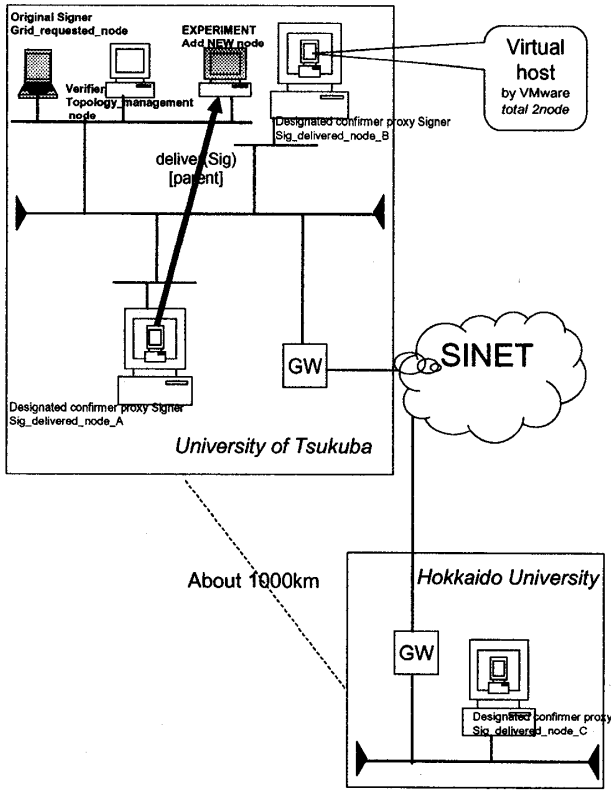


図 4　実験系

実証実験を実施するにあたり、筑波大学内に 3 つのセグメントと北海道大学内の 1 つのセグメントの合計 4 つのセグメントを構築した。なお、筑波大学と北海道大学の間は、SINET を経由して接続されている。各 GRID ノードは、Pentium3 1GHz の 1CPU で構成された IA PC であり、

Linux RedHat が動作する。さらに、その上で VMware を動作させ、さらにもう1ノード Linux RedHat が動作する環境を用意した。すなわち、合計 6 ノードが既に動作している GRID ノードとなる。さらに、筑波大学内の別のノードに同様に Linux RedHat が動作する IA PC を配置し、Topology management node とした。オリジナルの署名者となる GRID リクエストを出すノードについても同様に、同一のセグメントに接続した。リクエストは、実験の遂行上スクリプトファイルに記載して実行するようにした。

5.2.2 実験シナリオ

実証実験では、Topology management node と同じセグメントに新しい GRID ノードを追加し、証明書手続きを経て新たに GRID 環境に参加するための準備が完了するまでの時間を測定し、どれだけ証明書配布の手続きによる負荷が減り、またどのような影響があるかを評価することを目的として実験シナリオを定義した。なお、本実験を評価する目的は、証明書配布の時間を測定することであるため、オリジナルの GRID リクエストとしては、各 GRID ノードでループしてファイルに書き出すジョブを MPI ベースで記述し、ジョブを投入した。すなわち、ファイルを書き出すだけの GRID プログラムであり、終了条件は特に記述してはいない。詳細を以下に述べる。

1. 新たに追加するノードの初期情報として、筑波大学内にあるセグメント A のノード 1（VMware ではないホスト側ノード）を Parent_node と定義する
2. 各 GRID ノードから Topology_update()メソッドは、10 秒ごとに発行しトポロジマップ情報が更新される
3. 新たなノードをネットワークに接続した時点からの時間を測定した。測定精度を高めるために、各 GRID ノードは NTP サーバを利用して時間同期をとっている。
4. 新しいノードが追加されてから、オリジナルのリクエストを出す GRID_requested_node に準備完了のメッセージを発行するまでの時間を測定した。実験データの信憑性を考慮するために、合計 10 回実験を実施した。

5.2.3 実験結果

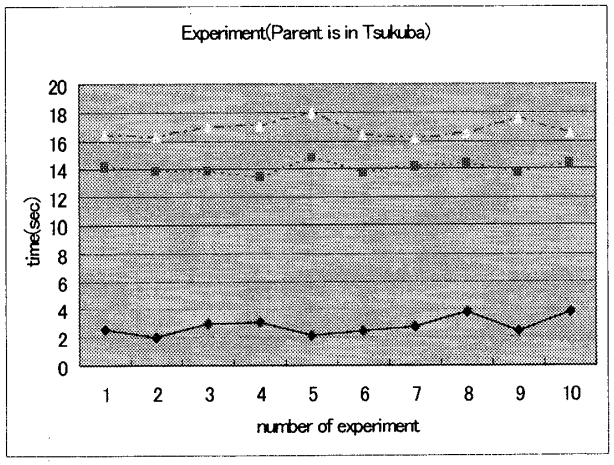


図 5 実証実験の結果

実証実験の結果を図 5 に示す。図 5 の縦軸は、時間(sec)を示し、横線は実験番号を示す。表中の実線は、新しいノードを追加し、Topology management node がその更新通知を受け取るまでの時間を示す。点線は、その新たに追加されたノードが親ノードから証明書を受け取るまでの時間を示す。一点鎖線は、証明書を受け取った後に検証者である Topology management node との間で検証作業が行われ、GRID_requested_node が準備完了であることの通知を受け取るまでの時間を示す。

この結果から、各 GRID ノードはトポロジマップ情報の更新するための時間として 10 秒間かかっていることを勘案し、その時間を差し引いた時間としても約 4〜5 秒程度で新たに GRID 環境に追加されたノードの検証が行われ、GRID 計算の準備が完了したことが示された。従来、GRID 環境においては、新たなノードを動的に追加することについてはほとんど言及されておらず、一般的には証明書を再度、CA に依頼して検証作業を進めるにあたり、オリジナルの署名者の負荷もさることながら、その手続きが膨大になることが考えられる。本プロトタイプでは、実証実験の都合上、一部実装を簡略化しているものの、実際に提案した手続きは全て行われている。これにより、提案した手法が実際のネットワーク上に適用することで、数秒程度で GRID 環境に動的にノードを追加することが可能であることを実証実験により確認することが出来た。

6. まとめと今後の課題

本論文では、GRID の計算ジョブなどのリクエスト処理が進められている環境において、動的に新たなノードを追加する際に、そのノードの信頼性を保証するために、GRID 環境におけるセキュリティのあり方という視点から、Designated Confirmer Proxy signature 手法を拡張した新たな手法を提案した。さらに、その評価のためのプロトタイプを実装し、実証実験を実施し、提案した手法の有効性を確認した。

最後に、今後の課題ではあるが、新たなノードを別のネットワークに追加した際にどの程度の負荷がかかるか、また、より具体的な GRID 計算環境において、本手法がどの程度有効であるかを評価するための実証実験を再度実施する予定である。

参考文献

[1]Globus Project, http://www.globus.org/
[2]M.Mambo, K.Usuda, and E.Okamoto, "Proxy signature: Delegation of the Power to Sign Messages", IEICE Transactions, Fundamentals, vol.E79-A, no.9, pp.1338--1353, 1996.
[3]M.Mambo, K.Usuda, and E.Okamoto, "Proxy Signatures for Delegating Signing Operation", Proceeding 3rd ACM Conference on Computer and Communications Security (CCS96), ACM Press, pp.48--57, 1996.
[4]D.Chaum, "Designated Confirmer Signatures", Lecture Notes in Computer Science 950, Advances in Cryptology-Eurocrypt94, Springer-Verlag, pp.86--91, 1995.
[5]T.Okamoto, "Designated Confirmer Signatures and Public-Key Encryption are Equivalent", Lecture Notes in Computer Science 839, Advances in Cryptology-Crypto94, Springer-Verlag, pp.61--74, 1994.