広域計算システム Ninf におけるユーザ認証

中 田 秀 基† 松 岡 聡††
佐 藤 三 久††† 関 口 智 嗣 †

昨今の急速なネットワーク技術の発展により分散高性能計算が可能となり、これを支援するソフトウェアフレームワークとしてわれわれの Ninf を含む幾つかのシステムが提唱されており、技術的にも成熟しつつある。しかし、この種のテクノロジの普及に必要とされるユーザの制限や計算機使用量に応じた課金、データの機密性保持といったシステムの社会的側面の研究は途上にある。本稿ではこの種の要請に答えるためのセキュリティ機構の一部であるユーザ認証をとりあげ、分散高性能計算に要請される認証機構について論じる。認証の強度とシステム利用の簡便性はトレードオフの関係にあり、システムの使用形態に応じて複数の認証機構を使い分けることが重要である。

Authentication for Ninf: Global Computation System

Hidemoto NAKADA,† Satoshi MATSUOKA,††
Mitsuhisa SATOH††† and Satoshi SEKIGUCHI †

Rapid growth of network technology made high-performance distributed computing possible, Technical aspects of software framework for high-performance distributed computing are already almost established. However, from social aspect, some important issues still remain open, i.e, access control or accounting. In this paper, we discuss authentication mechanism which is needed for the above issues. Strictness of authentication and easiness of system usage are tradeoff. Authentication machanism have to be choosen according to system usage.

1. はじめに

昨今の急速なネットワーク技術の発展により、地理的に分散して存在する計算資源を用いた分散高性能計算が可能になりつつある。分散高性能計算を支援するソフトウェアの枠組みとして、われわれの Ninf[8) を含む幾つかのシステムが提唱されており、純粋に技術的な意味では分散計算システムはほぼ完成しつつあると言える。

一方、この種のテクノロジを実際に普及させるためには、ユーザの制限や計算機使用量に応じた課金といったある意味で社会的な問題を解決していかなければならない。計算の性質によっては、ネットワークを通過する演算データそのものに対しても機密性が必要になる場合も考えられる。これらの要求を満たすためには、分散計算システムに何らかのセキュリティ機構を導入する必要がある。

一般に、インターネットのような公共のネットワークを通信に用いる場合、盗聴 (Eavesdropping)、偽造 (Forges)、乗っ取り (Hijack) 等の攻撃を受ける可能性がある。これらの危険に対処するには、認証 (Authentication) と通信内容の秘密化 (Privacy) が必要とされる。本稿ではこのうちの前者をとりあげ、分散高性能計算システムに要請される認証機構について論じる。

認証の強度とシステム利用の簡便性は一般にトレードオフの関係にある。認証を強固にすればするほど、システムセットアップ時やシステム使用時に余分な手間が必要となる。したがって分散システムの運用形態に応じて、どの程度の強度の認証が必要とされるのかを検討し、その要請に応じて認証の手法を選択する必要がある。

2 章で、認証機構一般に関してサマライズする。 3 章で、分散計算システムの認証機構への要請をまとめる。4 章で、 Ninf システムにおける認証機構を提案する。 5 章で、関連する他システムを紹介する。

2. 認証機構

認証は一般に、通信の両端が何らかの秘密 (シークレット) を持ち、それを持っていることを何らかの手法で互いに証明することで行われる。認証機構には大別して、パスワードによるもの、使い捨てパスワードによるもの、証明書によるものがある。

2.1 パスワードによる認証

パスワードによる認証はシークレットそのものを提示

† 電子技術総合研究所 Electrotechnical Laboratory
†† 東京工業大学 Tokyo Institute of Technology
††† 新情報処理開発機構 Real World Computing Partnership

することで認証を行っていると考えられる。これは最もシンプルだが、盗聴に弱いという大きな欠点がある。

2.1.1 rsh/rlogin の認証

rsh/rlogin の認証は、 Unix 環境では最も身近に用いられている認証機構である。ユーザ名、パスワードをネットワーク経由で転送し、コンソールからのログオンと同様に認証を行う。この際パスワードが平文でネットワークを流れることになり、非常に危険である。

また、サーバ側のファイルに信頼するクライアントホスト / ユーザを登録しておき、そのクライアントホスト / ユーザからの接続を許す機構も用意されている。クライアントホストの認証には IP アドレスを用い、ユーザの認証は基本的にクライアントプログラムを信じる。この際、クライアント側のポートがスーパユーザ権限でしか用いることのできない priviraged port を用いていることを求める。これによって、クライアントのスーパユーザを信じるのであれば、クライアント側のユーザ名が詐称されていないと信じることができる。

この認証機構は、ソケットの IP アドレスをクライアントマシンのアイデンティティとして用いるが、 IP アドレスの詐称は比較的容易であるため安全性には大きな問題がある。とはいえ、ユーザにかかる負担は少ない。

2.2 使い捨てパスワードによる認証

使い捨てパスワードによる認証は、シークレットそのものでなくシークレットから生成された二次的なものを送信することで間接的にシークレットを持っていることを証明する。

2.2.1 S/Key

S/Key[1] ではシークレットからある手順で連続的に生成される一連のパスワードをそれぞれ一度づつだけ用いる。この手順は n から n+1 を作るのは簡単だが、 n から n-1 を作るのは難しい手順となっている。使用時には生成されたのと逆順にもちいるので、途中のパスワードを盗聴しても、その次に用いられるパスワードを予測することは難しい。

2.2.2 同期式使い捨てパスワード

同期して変更されるパスワードの系列をクライアントとサーバが共有し、その時点で有効なパスワードを用いて認証を行う方法。この方法ではパスワードを盗聴されても問題は生じない。実際にはクレジットカードサイズの「トークン」と呼ばれるパスワード生成器をユーザが持ち歩き、それが表示する「現在のパスワード」をつかって認証を行う。

2.2.2.1 チャレンジ＆レスポンスによる認証

この手法では、接続されるサーバからクライアントへ「チャレンジ」と呼ばれる毎回異なる情報を送る。クライアントはこのチャレンジにシークレットを用いた計算を行い、その結果をレスポンスとしてサーバに返す。サーバ側でも同じ計算を行い、これがレスポンスと一致すればシークレットの共有が確認でき、認証ができたことになる。

これらの手法では、シークレットがネットワークを移動しないので、パスワードによる認証よりも安全であるが、何らかの方法で最初にシークレットを交換する必要がある。

2.3 証明書による認証

この手法では、信頼できる第三者機関を想定する。通信の両者はそれぞれ第三者機関から証明書の発行を受け、通信時にはこれを交換することで、互いを認証する。通信の両者が直接互いのシークレットを持つ替りに、第三者機関との間でシークレットを交換する。このため、初めて通信する相手であっても認証を行うことができる。

2.3.1 Kerberos

Kerberos[2] は、 MIT の Athena 計画の一部として開発された認証システムで、対称暗号のみを用いている。 KDC(key distribution center) と呼ぶ第三者機関にすべての参加者の対称鍵を保管している。

クライアントがサーバにアクセスする際には、 KDC からセッションキーをクライアントの鍵で暗号化したものと、クライアントの素姓とセッションキーをサーバの鍵で暗号化したもの (チケット) を受け取り後者をサーバに送る。これでセッションキーが共有でき、同時に KDC が保証するクライアントの素姓もサーバに伝わる。

Kerberos は非常に巧みで強固な認証システムだが、対称暗号を用いているため KDC にすべてのユーザの秘密鍵が集中することになり、 KDC が破られた場合の損害が非常に大きいことが問題点とされている。

2.4 SSL

SSL(Secure Socket Layer) は、 Netscape Communication 社が提唱している暗号化プロトコルで、 Netscape Navigator 等に組み込まれており、 WWW における事実上の標準プロトコルとなっている。その名の通りソケットレイヤを代替する形で使用できるので、さまざまなアプリケーションで使用することができる。

SSL は公開鍵暗号を用いた証明書による認証と、セッションごとに生成される対称鍵暗号による暗号化を行う。認証は階層的な証明書発行機関の存在を前提にしている。この場合の証明書は、通信者の公開鍵を発行機関の秘密鍵で暗号化したものである。送られてきた証明書を発行機関の公開鍵で復号すれば通信相手の公開鍵を入手できる。その公開鍵を用いてセッション鍵を交換して、通信を行う。公開鍵暗号は暗号化 / 復号化に多大な計算コストがかかるが、通信自体の暗号化には比較的計算コストの軽微な対称鍵暗号を用いているので暗号化のコストはそれほど問題にならない。

公開鍵暗号を用いているため証明書発行機関には公開鍵のみが蓄えられるため、秘密鍵が蓄えられる Kerberos の KDC と異なり、大きなセキュリティボトルネックにはならない。また、証明書発行機関を階層化することもできる。上位の発行機関が発行機関を認証して証明書を

発行する。下位の発行機関による証明書には下位の発行機関自身の証明書も添付する。すると上位の発行機関の公開鍵を持っていれば、下位の発行機関の公開鍵を取得でき、下位の発行機関が出した証明書を復号することができる。

メイルシステム PEM(Privacy Enhanced Mail) も同等の認証機構を用いている。

2.4.1 PGP

PGP(Pretty Good Privacy)[3] は、 Phil Zimmermann によって開発された公開鍵暗号を用いた暗号化 / 署名システムである。 PGP の認証機構は集中管理された「信頼できる第三者機関」は用いない。個々のユーザが自分の信頼できるユーザを保証する、という形で信頼の和を作り上げ、それが認証機関と同等の効力を発揮する。

第三者機関による証明書の発行を待たずに使用を開始できるため、使用開始時の障壁が低いが、信頼の和をどの程度信頼していいのかが問題で、高度なセキュリティを要求される場合には運用が難しいと考えられる。

3. 分散計算システムにおける認証

3.1 分散計算システムの概要

一般に分散計算システムは以下の要素からなる。

- クライアント
- 計算サーバ
- リソースアロケータ
- リソースモニタ
- リソースデータベース

リソースモニタが定期的にリソースの状態をモニタし、リソースデータベースに登録する。クライアントが計算を行うには、まずリソースアロケータにアクセスする。リソースアロケータは、データベースから情報を取り出し、適切なリソース (サーバなど) をクライアントに紹介する (図 1)。

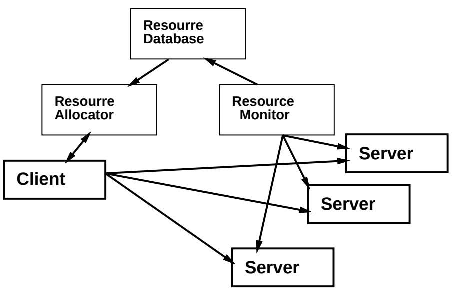

図 **1** 分散計算システム

3.2 分散計算システムの運用形態

分散計算システムにはさまざまな運用形態が考えられる。大きく分けて次のように分類することができる。それぞれにおいて、要求される認証強度のレベルは異なる。

- サイト内分散計算
  たとえば一つの研究室などの単一の管理ポリシで管理されている範囲内だけで使用する。ユーザの人数は数十人程度、計算リソースも数十台程度で、比較的容易に集中的に管理することができる。
  認証機構にはそれほどの強度は必要とされないので、使用の簡便さを重視するべきである。
- キャンパスワイド分散計算
  たとえば一つの大学など、複数のそれぞれ異なる管理ポリシで管理されているサイトが含まれてはいるが、個々の参加者を物理的に特定することが可能で、各サイトの善意を信用できる環境。ユーザの人数は数百人程度、計算リソースは数百台程度で、集中的に管理することはむずかしい。
  認証機構にはある程度の強度が要求される。
- グローバル分散計算
  複数のそれぞれ異なる管理ポリシで管理されているサイトが含まれていて、それぞれのサイトの管理者は信用できるが、個々のユーザは信頼できない。ユーザの人数、計算リソースとも千を越え、集中的な管理は現実的ではない。階層的な管理構造の導入が不可欠である。
  認証機構には相当程度の強度が要求される。

分散システムにおける認証システムはこれらの運用形態に則して柔軟に運用できるものでなければならない。

3.3 分散計算システムの認証機構への要請

分散計算システムの認証機構は以下の条件をみたしていなければならない。

- リソースアロケーションシステムとの連動
  リソースアロケータによって紹介されたシステムに実際にアクセスしてみたら、認証で拒否されたというようなことがないようにしなければならない。
- 階層化可能
  個々のサイト内の管理はサイト内で完結していることが望ましい。したがって広域での管理は階層的に行う必要がある。
- 拡張可能
  始めはローカルに構築したシステムが、連続的に外部に拡張していくことができることが望ましい。

4. Ninf システムの認証機構

4.1 Ninf システムの概要

Ninf はクライアント / サーバをベースとた分散計算システムで、クライアントプログラマは、サーバ側のルーチンをクライアント側のルーチンと透過的に使用することができる。 Ninf は リソースデータベースを用いたリソースアロケーション機構を持つ[9]。

4.2 Ninf の認証機構

Ninf システムで用いるために、前章で述べた要求を

満たす認証機構を設計した。この認証機構は以下の特徴を持つ。

- ローカルとグローバルとで別の認証機構を用いる。
- ローカルな範囲で使用を開始し、連続的にグローバルなシステムと連結できる。
- リソースアロケータと連動する。

プロテクトされたローカルなサイトの内部での認証機構としては、 rsh 相当の機構を用いる。すなわち、ホスト名は IP アドレスを信じ、ユーザ名は特権ポートの使用を前提に自己申告を信じる。認証の強度は弱いが、ユーザに存在を意識させずに使用させることができ、使用開始の障壁が低い。

グローバルな空間での認証機構には十分な認証機構と暗号化機能を持つ SSL を用いる。 SSL レベルでの認証の単位は、ユーザではなくサイトとする。 SSL は使用開始時に証明書を取得するという手間がかかるが、認証単位をサイトとすることで、個々のユーザへの負担が低減される。

アクセス制限にはアクセスコントロールリストを用いる。アクセスコントロールリストの対象はサイト単位でもよいし、ユーザ単位でもよい。サイトより上位のドメインに対して一括してアクセス権を設定することも可能である。コントロールの対象となるリソースは、サーバ単位でもよいしライブラリ単位であっても良い。

4.2.1 ローカル使用時の認証機構

図 2にローカルに使う場合の認証機構を示す。

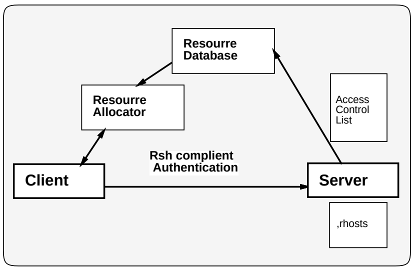

図 **2** ローカル使用時の認証機構

各サーバはアクセスコントロールリストを持ち、その内容をリソースデータベースに登録する。クライアントは接続時にリソースアロケータ経由でデータベースにアクセスして、サーバの紹介を受ける。クライアントがサーバにアクセスすると、サーバは rsh と同様にそのユーザの.rhost を参照して認証を行い、さらにアクセスコントロールリストを参照して、接続の可否を決定する。

認証には rsh と同じ機構を用いるので、個々のユーザが新たに設定を行う必要はない。

4.2.2 広域使用時の認証機構

図 3に広域環境での認証機構を示す。ローカル時の認証機構はそのまま用い、さらにプロキシを導入することで複数のサイトに拡張を行う。サイト間の通信はプロキシを介して行い、プロキシ間の通信は SSL で認証する。キャンパスワイドのように、通信自体を暗号化する必要がない場合は、 SSL の認証だけを用い通信の暗号化は行わない。

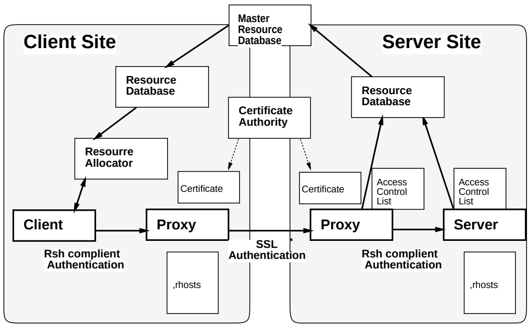

図 **3** 広域環境での認証機構

各サイトのプロキシは共通の証明書発行機関から証明書を取得する。キャンパスワイドの場合は独自に発行機関を作成すればよいし、グローバルの場合には我々が用意する発行機関を利用する。さらに各サイトに Ninf 用のアカウントを一つ用意し、プロキシをそのアカウントで実行する。サーバを提供する場合、プロキシにもアクセスコントロールリストを用意する。

クライアントが他のサイトのサーバに通信する際には、まず自分のサイトのプロキシに対して接続する。プロキシへの接続は rsh 相当の認証で行うので、プロキシには接続してきたユーザがわかっている。プロキシは、サーバサイトのプロキシに SSL を用いて接続し、使用を試みているユーザ名を通知する。この際にアクセスコントロールリストを用いてアクセス制限を行う。ここで制限されなかった場合、サーバサイトのプロキシはサーバに接続する。このとき rsh 相当の認証を行う。プロキシは Ninf 用のアカウントで実行されているので、サーバはプロキシからの接続であることがわかる。サーバはプロキシからクライアントの素姓に関する情報を受け取り、ローカルなアクセスコントロールリストを参照して接続の可否を決定する。

この手法の特徴は、ローカルに Ninf を使用している状態から、連続的にグローバルな分散計算に参加できるようになる点である。クライアント側では、証明書を取得してプロキシを設定するだけでよく、個々のユーザには全く負担はない。サーバ側では、アクセスコントロールリストをグローバル用に拡張する必要があるが、基本的にはプロキシのアクセスコントロールリストだけで行うことができるので、個々のサーバのリストを変更する必要はない。

4.3 試 験 実 装

各認証手法の問題点と性能を検討するために、 Ninf システムに rsh と同じ手法による認証機構と、 SSL を

用いた認証機構を試験的に付加した。今回は、両機構ともサーバとクライアントのみに実装しており、リソースアロケータなどには実装していない。

4.3.1 rsh 相当の認証

先に述べたように、 rsh の認証法においてはクライアント側が、スーパユーザしか使えないポートを使うことが重要である。しかし、 Ninf のようなシステムのクライアントプログラムをスーパユーザ権限で動かすことを期待するのは非現実的である。

そこで、スーパユーザに setuid されたトンネル機能を持つコマンドを作成した。このプロセスは起動されると親プロセスのユーザ ID を取得して宛先の Ninf サーバに通知し、その後親プロセスからの出力をフォワードする。このコマンドをクライアントライブラリが起動することでユーザに意識させることなく、 rsh と同等の認証機構を提供することができる。この構造では、すべてのデータがこのプロセスで中継されるため、認証のコストの他に通信自体にもオーバヘッドが生じることが予想される。

4.3.2 SSL による認証

Eric Young、 Tim Hudson による SSL のフリーの実装である SSLeay[10] を用いて、 SSL による認証を実現した。 SSL はトランスポートレイヤで暗号化を行うので、基本的に、 Unix で用いる標準的なソケット関係のシステムコールを、 SSLeay が用意しているライブラリ関数で置き換えるだけで SSL を用いることができる。

SSL の認証は、双方から証明書を送り、それを検証することで行う。これにはそれなりの計算量が要求される。またデータ転送を暗号化する場合には当然そのコストがかかる。

4.3.3 試験実装による実験

分散計算システムで認証を行う場合のコストを検証するために、上述の試験実装を用いて認証と通信におけるそれぞれのオーバヘッドを測定するために実験を行った。

実験は以下の環境で行った。

- クライアント: Ultra SPARC 167MHz
- サーバ: DEC Alpha 333MHz
- ネットワーク:100base/T
- プログラム: linpack (non-block)

行列サイズは 500, 700 とした。それぞれ転送されるバイト数は 2Mbyte, 4Mbyte 程度である。

認証なしの場合と、 rsh 相当の認証を行う場合、 SSL による認証を行う場合の 3 通りの実験を行った。表 1にその結果を示す。プロトコルコストは、通信、計算以外にかかった時間であり、プロトコルコストの増分が、認証によるオーバヘッドであると考えることができる。

認証のコスト、すなわちプロトコルコストの増分は、両手法とも十分少なく、無視できることがわかる。

通信におけるオーバヘッドは、 rsh 相当では一つ余計にプロセスを経由しているが、問題になるほどのオーバヘッドはない。 SSL では、それなりに大きなオーバヘッドが生じている。このオーバヘッドと通信速度の関係を調べるために、 WAN を用いた実験を行った。

表 1 LAN におけるプロトコルコストと通信時間 (秒)

サイズ		プロトコルコスト	通信
500	認証なし	0.12	0.28
	rsh 相当 (増分)	0.26 (0.14)	0.29 (0.01)
	SSL(増分)	0.18 (0.05)	3.14 (2.86)
700	認証なし	0.13	0.55
	rsh 相当 (増分)	0.23 (0.10)	0.57 (0.02)
	SSL(増分)	0.18 (0.05)	5.90 (5.35)

サーバには上述のものを用い、新たにクライアントを地理的 / ネットワーク的に離れたサイトに設置した。両者の間の ftp による通信スループットは、 80KByte/s 程度である。結果を表 2に示す。

表 2 WAN における通信時間 (秒)

サイズ		通信
500	認証なし	12.9
	SSL(増分)	13.1 (0.2)
700	認証なし	26.4
	SSL(増分)	26.1 (-0.1)

700 の時に逆に通信時間が短くなっていることからもわかるように、ネットワークの状態が不安定で測定誤差が大きくなってしまっているが、通信における暗号化のオーバヘッドが測定誤差の範囲内までに見えなくなっていることがわかる。これは、通信速度が十分遅いため、暗号化 / 復号化の計算と通信とがオーバラップして実行されるためであると思われる。したがって WAN 環境では SSL による暗号化のコストは、ほぼ無視できると言える。

5. 関 連 研 究

5.1 Globus

Globus[4] は、広域計算システムのためのツールキットであり、通信ライブラリやディレクトリサービスやリソース管理機構など、比較的低位の機構を提供している。ユーザは Globus の上に実際に使用するシステムを構築する。 Globus は、認証機構として GSI(Globus Security Infrastructure) を提供している[5]。 GSI には、各サイトのローカルセキュリティポリシは変更しない、子プロセスへのアクセス権の伝搬が可能、といった特徴がある。

Globus にはグローバルな単一のユーザ名空間があり、これをサイトローカルなユーザ名空間にマップする。サイト内ではサイトローカルのセキュリティポリシにしたがってアクセスコントロールが行われる。ローカル空間へのマッピングの際に、 Globus が運用する発行

機関による証明書にもとずいた認証が行われる。

Globus は、リモートプロセスのフォークをベースにしたシステムなので、ユーザが持つアクセス権を次々にフォークしたプロセスに伝搬していくことが重要である。 Globus では個々のプロセスが証明書をもつ。ユーザがプロセスをフォークする際には、自分の証明書で証明した証明書をそのプロセスに与える。同様に親プロセスが子プロセスをフォークする際には、親プロセスが子プロセスを証明する。それぞれのプロセスの証明書には、すべての親プロセスの証明書が残るので、各リソースは証明書を遡りプロセスの元のユーザを知ることができ、そのユーザに対するアクセス権を個々のプロセスに対して設定することができる。

問題点としては、個々のユーザを必ずグローバルな名前空間に登録して、発行機関から証明書を取得しなければならず、手続きが煩雑である点が挙げられる。また、システムの使用に先行して Globus へのログオン手続きが必要であり、この点も煩雑である。

5.2 Legion

Legion[6] はオブジェクトをベースとした分散計算システムである。 Legion ではすべてのオブジェクトが認証の対象となる主体となりうる[7]。それぞれのオブジェクトは個々のオブジェクトに対して、メソッド単位でアクセスを制御することができる。オブジェクト間のメッセージパッシングには、呼び出されるオブジェクト以外に、 CA(Calling Agent)、 RA(Responsible Agent)、 SA(Security Agent) の 3 つのオブジェクトが介在する。 CA が呼び出し主体であり、 RA はその呼び出しが含まれる一連の呼び出し全体を行っている主体で、ユーザ ID の拡張ととらえることができる。 SA はセキュリティポリシを実現するものである。アクセス権は、 CA に対してでなく RA に対して設定されるので、 RA が持つアクセス権が伝搬していくことになる。

あるオブジェクトが他のオブジェクトのメソッドを呼び出す際には、それに先行して`MayI` メソッドが呼び出される。呼び出される側のオブジェクトは、呼び出すことを許すメソッドのリストを含んだ「ライセンス」を発行する。実際のメソッド呼び出し時にはこのライセンスを用いる。

Legion のセキュリティ機能は非常に強力だが、その反面、多大なオーバヘッドが生じることが予測される。

6. おわりに

本稿では、課金やユーザ制限のベースとなる技術として、高性能分散システムのための認証機構について論じた。認証機構の強度と使用の容易さはトレードオフの関係にあり、単一の最良解があるわけではない。用途に応じて幾つかの手法を組合わせ、使い分けていくことが重要である。われわれは、強度は弱いが使用は容易な認証と、強度が強いが手間のかかる認証手法を組み合わせることで、ローカルには使用を開始しやすく、危険な外部ネットワークでは十分な強度の認証を行う認証機構を提案した。

今後の課題としては、今回提案した認証システムの実装を行い、実際に使用に供することで、実用上の問題点を洗い出すことが挙げられる。また、 Globus や Legion ではサポートしているアクセス権の伝搬についても導入する必要がある。さらに、他の分散システムとの連携も検討しなければならない。具体的には Globus ツールキットを用いて Ninf を実装することを検討している。

参考文献

1) : ftp://ftp.bellcore.com/pub/nmh.
2) Kohl, J. and Neuman, C.: The Kerberos Network Authentication Service (V5), RFC 1510.
3) Garfinkel, S.: PGP 暗号メールと電子署名, O'REILLY (1996).
4) Foster, I. and Kesselman, C.: Globus: A metacomputing infrastructure toolkit., *Proc. of Workshop on Environments and Tools, SIAM.* (1996).
5) Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S.: A Security Architecture for Computational Grids, *Proc. 5th ACM Conference on Computer and Communication Security, toappear* (1998).
6) Grimshaw, A. S., Wulf, W. A., French, J. C. and Alfred C. Weaver, P. F. R. J.: Legion: The Next Logincal Step Toward a Natiowide Virtual Computer, CS94-21, University of Virginia (1994).
7) Wulf, W.A., Wang, C. and Kienzle, D.: A New Model of Security for distributed Systems, CS 95-34, University of Virginia (1995).
8) Sato, M., Nakada, H., Sekiguchi, S., Matsuoka, S., Nagashima, U. and Takagi, H.: Ninf: A Network based Information Library for a Global World-Wide Computing Infrastracture, *Proc. of HPCN'97 (LNCS-1225)*, pp. 491–502 (1997).
9) 竹房あつ子, 中田秀基, 合田憲人, 小川宏高, 松岡聡, 長嶋雲兵: Ninf システムにおけるジョブスケジューラの実装と予備的評価, 情報処理学会研究報告 HPC, Vol. 98, No. - (1998).
10) Young, E. and Hudson, T.: CryptSoft Pty Ltd. Home Page, http://www.ssleay.org/.