マルチメディア通信と分散処理 71−15
（１９９５．７．１３）

分散システムにおけるユーザ認証システムの実現

杉山広幸　小畠雅典　田辺克弘

NTT　ヒューマンインタフェース研究所

分散システムが一般的になるにつれ、新たなセキュリティの問題が顕在化しつつあり、不正なユーザの侵入を防ぐユーザ認証の機能が求められている。今回、共通鍵暗号を用いた信頼できる第三者に基づくユーザ認証プロトコルを設計、実装した。実装においては、インターネットで提案されているGSS-APIをセキュリティ機能を提供するインタフェースとして採用した。本稿では、認証プロトコルの設計とその実装について述べる。

Design and Implementation of User Authentication System in Distributed Systems

Hiroyuki Sugiyama　Masanori Obata　Katsuhiro Tanabe

NTT Human Interface Laboratories

Security become actual problem with popularization of the distributed systems. So, authentication system for preventing intruder is required. We design and implementation authentication protocol using symmetric encipherment algorithms based on the model proposed by Needham et al. Furthermore, we adapt Internet GSS-API as an application program interface for present security facilities. This paper describe the design of proposed authentication protocol, and its implementation.

1．はじめに

クライアント・サーバ・システムのようなネットワークで接続された分散システムが、従来の汎用機を用いた情報システムに置き換わり、広く普及しつつある。このような分散システムでは、情報の共有化が容易になるなどの利便性をもたらすとともに、機密への不正アクセスが容易になるなどの新たなセキュリティ上の問題が顕在化しつつある。

分散システムにおけるユーザ認証は、ユーザ名とパスワードを用いたユーザ認証が一般的に採用されており、パスワードがそのままネットワーク上を流れている。盗聴したパスワードを用いて正規ユーザになりすまし、サーバに不正にアクセスすることは侵入者にとっては容易なことである。このような脅威に対しては、暗号技術に基づくより安全なユーザ認証の機構が必要である。

そこで、分散システムにおいてユーザ認証を提供するために、共通鍵暗号を採用したユーザ認証プロトコルを設計し、ユーザ認証システムとしてOSF/DCE上に実装、さらに本システム上で動作するアプリケーションとしてセキュアなファイル転送システム(FTP)を実現した。本稿ではシステムの設計方針と実装、ならびにその評価について述べる。

2．背景

2．1　過去における研究

これまで、分散システムにおける暗号技術に基づくユーザ認証方式については多くの研究者によって提案されている。特に、Needham等[1]は共通鍵暗号、公開鍵暗号の両者について信頼できる第三者(TTP: Trusted Third Party)の存在を前提とするユーザ認証方式を提案している。これらの研究成果は標準化の対象となり、ISO/IEC　SC27で作業が進められており、ISO/IEC-9798シリーズ[2]として文書化されている。

このような、認証プロトコルの研究成果を受け、実際に実装まで行った研究としては MIT の Athena（アテナ）プロジェクトの Kerberos （ケルベロス）[3][4]が、よく知られている。Kerberos では Needham 等による共通鍵暗号と信頼できる第３者の存在を前提とする認証プロトコルをベースとして、メッセージの再送によるなりすまし防止のためにタイムスタンプを用いている。

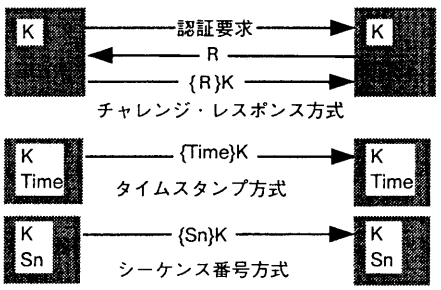

図1　再送攻撃の防御方式

現在はバージョン５β５がフリーソフトウェアとして配布されており、大学などを中心として広く利用されている。また、幾つかのベンダーが商業的にサポートを行っている。

しかし、kerberosはあくまでもMITにおける学内情報システムの一部として利用されることや、実装ハードウェアがUNIXワークステーションであることを想定し設計されており、現在の企業における、クライアントがパソコンで、さらにISDNを含む多様なネットワークで接続されたような情報システムに適応することを考えた場合、以下に示す問題がある[5]。

最初に、Kerberosでは再送攻撃に対する対策としてタイムスタンプを採用しているので、全てのクライアントマシン、サーバマシン間で時間が厳密に同期していることが必要である。しかし、公衆回線等でリモートアクセスを行うようなパソコンに対しても時間の同期を求めるのは極めて困難であり、現実的ではない。

さらに、パスワードによる個人認証を採用しているために、従来からUNIXで問題にされているパスワード推測攻撃に対しては無力である。

また、クライアントとしてシングルユーザで利用されるUNIXワークステーションを想定しており、クライアント上に安全に情報を格納できることを期待しているが、クライアントとしてパソコンを用いる場合は安全に情報を格納することはできない。

さらに、企業においてはその情報の重要さに応じたセキュリティ対策がとられていると考えられる。このような複数のセキュリティレベルに柔軟に対応したセキュリティ機能を提供できてない。

2．2　目的

本システムは、企業内情報システムとして用いられるマルチベンダーで構成されるクライアント・サーバ・システムに、以下の特徴を持つユーザ認証機構を提供することを目的とする。

1)時間同期に依存しない認証
2)クライアントの環境によらず安全な認証
3)多様なセキュリティポリシーのサポート
4)マルチベンダー環境におけるポータビリティ

3．設計方針

本システムは以下の方針に基づき設計した。

3．1　再送攻撃

ユーザ認証においては、クライアントとサーバ間でメッセージを交換し相互に認証しあう。このとき、侵入者がやりとりされるメッセージを蓄積し、あたかも正しいクライアントのようにそのメッセージをサーバに対して送ることによってクライアントに成りすまそうとする再送攻撃の危険がある。

このような再送攻撃を防ぐためには、メッセージの一意性、等時性を保証しなければならず、対策としては、以下の３つの方式がよく知られている（図1）。

1．チャレンジ・レスポンス方式
2．タイムスタンプ方式
3．シーケンス番号方式

タイムスタンプ、シーケンス番号方式では、共にあらかじめ時間、シーケンス番号などの情報を共有しなければならないが、お互いに信頼関係のないエンティティ間でこれらの情報を安全に共有することは容易ではない。特に、常時接続されていない、もしくは、非常に遅延のある通信路で接続されているパソコンにおいて安全に行うことは困難である。

さらに、各エンティティはこれらの値を、秘密鍵と同じように侵入者によって改竄されないように安全に守らなければならない。これらの値が改竄されれば再送攻撃を侵入者に許すことになる。

また、タイムスタンプ方式では、ある時間内に受け取ったメッセージを蓄積し、以降に受け取ったメッセージと比較しなければ完全に再送攻撃を防ぐことはできないが、一般にこのような実装は困難である。

チャレンジ・レスポンス方式はメッセージの交換が増えるが、環境に依存せず、それ自身で安全な方式である。

我々は、ISO/IEC 9798-2に示されている５パス認証方式に準拠するチャレンジ・レスポンス方式を本システムの認証プロトコルに採用し、公衆回線を用いたリモートアクセスで接続されるようなパソコンのように時間同期を実現することが困難、かつ時間の改竄が容易に行うことが可能な環境においても安全な認証を実現する。

３．２　個人認証方式

Kerberosで採用されているようなパスワードベースのユーザ認証システムでは、常にパスワード推測攻撃の危険にさらされている。さらに、クライアント上のファイルのセキュリティが期待できないパソコンなどにおいては、クライアント上のプログラムを置き換えることでユーザのパスワードを盗む「トロイの木馬」攻撃にも全く無力である。ICカードとパスワードを用いる「Two Factor Authentication」では、侵入者は単にパスワードを盗むだけではなく、ICカードを盗むか、ICカード中の秘密鍵を読みとり偽造しなければ正しいユーザになりすますことができない。

しかし、ICカードを用いた個人認証ではICカードそのものに加え、クライアントにICカードリーダが必要である。必ずしもここまで厳密な個人認証が必要でない場合もあるであろう。

我々は、ユーザ（エンティティ）の秘密鍵の管理方式を指定することで個人認証方式を選択可能とし、クライアントが信頼できない環境におけるICカードを用いた厳密な個人認証や、パスワードを用いた簡易な個人認証を選択できるようにする。

３．３　暗号方式

我々は、認証プロトコルや、メッセージの完全性、機密性保護のために暗号を用いる。

まず、実用的な応答速度を実現するために暗号方式としては公開鍵暗号ではなく共通鍵暗号方式を採用する。

さらに、一般にDES型の共通鍵暗号は段数や鍵長、暗号利用モードを選択することで暗号強度を変更し、暗号化、復号にかかる処理時間を選択できる。たとえば、NTTの開発した共通鍵暗号FEAL[6]では、段数として8、16、32などを、鍵長として8バイト、16バイトが選択可能である。従って、セキュリティのレベルによって利用する暗号方式を変更したいこともあるであろう。また、その実現方式としても、ソフトウェアのみならず高速化のために暗号処理専用ハードウェアを利用したい場合もあるであろう。そのうえ、米国の輸出規制にみられるように政治的な理由で利用できる暗号方式が制約される場合もあるであろう。このように、多様なセキュリティポリシーの実現やプラットフォームの採用においては、暗号方式やその実現方式を特定の方式に限定できない。

従って、我々は、認証プロトコルそのものを固有の暗号方式に依存しないようにし、かつ用いる暗号方式をクライアントとサーバ間でネゴシエーションする機構を実現することで、複数の暗号方式を選択可能とする。さらに、メッセージ保護においても、メッセージ中に保護方式を指定する領域を設けることで、複数の保護方式を指定可能とする。

これらの実現のために、暗号方式やその実現方式を指定する暗号機構識別子に従い、暗号化、復号、認証子の計算、鍵管理の機能を、固有の暗号方式、実現方式に独立して上位アプリケーションに提供する暗号インタフェースを内部インタフェースとして採用する。かつ、容易に暗号方式、実現方式が拡張可能とする。

さらに、メッセージ保護方式は、後で述べるGSS-APIを介してアプリケーションから指定できるようにする。

３．４　アプリケーションプログラグインタフェース（API）

一般にユーザ認証システムの導入にあたっては、アプリケーションをそのユーザ認証システム対応に修正する必要がある。Kerberosにおいてもアプリケーションを KerberosのAPIを呼び出すように変更しなければならない。アプリケーションを個々のユーザ認証システム毎に対応させなければならず、ユーザ認証システムの導入の妨げになっていた。このような問題に対する解として、IETFから標準セキュリティインタフェースとして GSS-API（Generic Security Service Application Program Interface）[8][9]が提案されている。GSS-APIは抽象化されたセキュリティインタフェースを上位アプリケーションに提供する。従って、このGSS-APIを呼び出すことで、アプリケーションを個々の認証機構に依存しないように実現することが可能である（図2）。

我々は、アプリケーションのポータビリティを実

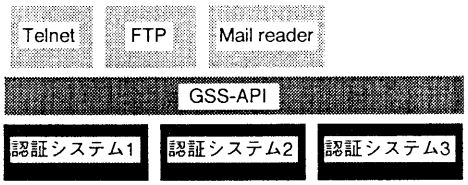

図2　GSS-API

現するために、APIとして標準のセキュリティインタフェースであるGSS-APIを採用する。

GSS-APIでは、まず証明書を獲得し、その証明書に基づき、サーバ間とセキュリティコンテキストを確立することにより、ユーザ認証を行う。そして、このセキュリティコンテキストに基づき各種のメッセージ保護を行う。GSS-APIでは、証明書はアプリケーションの外側で、デフォルト証明書として、実装に固有な方法によって獲得されていることを勧めている。一般的に、デフォルト証明書の獲得は、個人認証のためにユーザとの対話をプラットフォームに固有のGUIで行うようなログインユーティリティなどのプログラムで行われる。認証システム自身のポータビリティを実現するためには、プラットフォームに固有なGUIなどはログインユーティリティで、デフォルト証明書の獲得などはGSS-APIが提供することが望ましい。しかし、GSS-APIはアプリケーションに対するインタフェースのみを規定しており、デフォルト証明書の獲得についてのインタフェースを規定していない。

そこで、我々は、ユーザ認証システム自身をポータブルとするために、デフォルトの証明書を獲得、解放するインタフェースをGSS-APIに追加し、これらの機能をGSS-API内部でサポートする。

3．5　マルチプラットフォーム

一般にユーザ認証システムではユーザ名、ユーザの秘密鍵などを管理する名前管理が必要となる。多くの認証システムでは、元々の分散システムが持つ名前管理とは独立に、名前管理を導入している場合が多い。そのため、元々の名前管理と認証システムの持つ名前管理の間の同期の問題を各分散システム毎に対応しなければならない。

OSFがリリースしている、RPCや分散ファイルシステムディレクトリサービスなどの機能を提供するDCE(Distributed Computing Environment)は、分散システムにおいてマルチベンダーによってサポートされ標準プラットフォームの地位を固めつつある。

我々は、本システムをマルチベンダー対応とするために、DCEをプラットフォームとして採用し、名前管理にもDCEの登録簿サービスを利用する。

4. 認証プロトコル

前節で述べた方針に従い設計した本システムの認証プロトコルの概要を示す。

4．1　前提条件

認証プロトコルは下記の前提条件のもと設計した。

1）クライアントは物理的に安全でない。
2）物理的に安全な信頼できる第三者が存在する
3）サーバ上の秘密鍵は安全に管理されている
4）ICカードの利用を可能とする
5）各エンティティ間は時間が同期していない
6）特定の暗号方式、その実現方式に依存しない

4．2　用語

本プロトコルで用いる用語について説明する。

証明書：クライアントが正当なクライアントであることを示す情報であり、クライアントとTGSで共有され、認証サーバから発行される。

トラスティ：サーバとの間で安全な通信を行うためにクライアントとサーバの間で共有され、証明書に基づきTGSから発行される。

認証サーバ：TGSからの証明書発行要求に従い、証明書を発行する。

トラスティ発行サーバ（TGS）：クライアントからの証明書要求に従い、認証サーバから証明書を取得し、クライアントと共有し、サーバからのトラスティ発行要求が証明書を共有するクライアントからの要求に基づくものであればトラスティをサーバに発行する。

4．3　プロトコル詳細

認証プロトコルは以下の５つのフェーズより構成する。

1）証明書獲得フェーズ
　クライアントとTGSで証明書を共有する
2）トラスティ獲得フェーズ
　既に獲得している証明書に基づきクライアントとサーバ間でトラスティを共有する
3）メッセージ保護フェーズ

表1　記号の意味

記号	意味
Nx	エンティティxの名前
Rx	乱数
Kx	エンティティxの秘密鍵
KSx-y	エンティティx、y間で共有するセッション鍵
ICx-y	クライアント証明書
ACx-y	TGS証明書
ITx-y	クライアントトラスティ
ATx-y	アクセプタトラスティ
{M}K	鍵Kで暗号化されたメッセージM

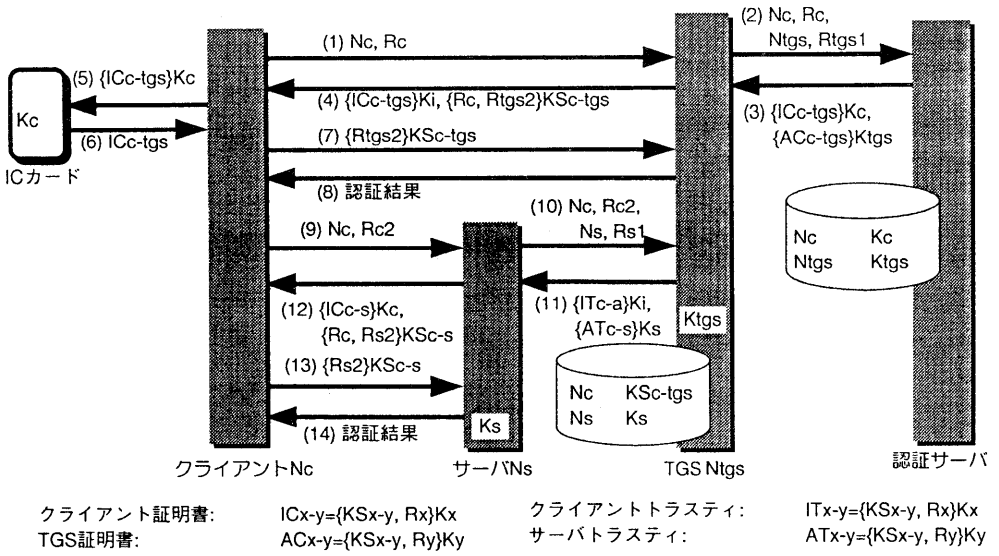

図3　認証プロトコル

既に獲得しているトラスティに基づきクライアントとサーバ間で安全にメッセージを交換する

4）トラスティ解放フェーズ
獲得したトラスティを解放する

5）証明書解放フェーズ
獲得した証明書を解放する

以降図3に従い、証明書獲得フェーズを例に認証プロトコルについて詳細に説明する。

a. クライアントは自分の名前Ncと乱数RcをTGSに送り証明書を要求する。

b. TGSはさらに自分の名前Ntgsと乱数Rtgs1を付加し、認証サーバに証明書を要求する。

c. 認証サーバは名前が登録されているならば、セッション鍵KSc-tgsを生成し、各々の秘密鍵で送られてきた乱数とともに暗号化して証明書として返す。

c. TGSはTGS証明書を自分の秘密鍵Ktgsで復号し、乱数Rtgsを取得したならば正しい認証サーバが発行した証明書であると判断し、セッション鍵KSc-tgsを取得する。次に、クライアント証明書ICc-tgsと、クライアントから送られてきた乱数Rc、新たに生成した乱数Rtgs2を暗号化したメッセージ{Rc, Rtgs2} KSc-tgsをクライアントに返す。

d. クライアントは、秘密鍵管理方式としてICカードを指定されていたならば、クライアント証明書をICカードに送り、ICカード中のユーザの秘密鍵Kcで復号し、乱数Rcを取得したならば正しい認証サーバが発行した証明書であると判断し、セッション鍵KSc-tgsを取得する。次に、メッセージ{Rc, Rtgs2} KSc-tgsをセッション鍵KSc-tgsで復号し、乱数Rcを取得したならば正しいTGSであると判断する。さらに、自身を証明するためにメッセージから得た2番目の乱数Rtgs2をセッション鍵KSc-tgsで暗号化したメッセージ{Rtgs2}KSc-tgsをTGSに送る。

e. TGSはメッセージをセッション鍵KSc-tgsで復号し、乱数Rtgs2を取得したならば正しいクライアントであると判断し、その結果をクライアントに伝える。

以上の手順によりクライアントとTGSは相互に認証し、さらに証明書を共有する。トラスティ獲得フェーズも証明書獲得フェーズと同様な手順によって行われ、クライアントとサーバ間でトラスティを共有できる。

5．実装

本システムは以下に示すプログラムより構成した（図4）。

1）GSS-APIライブラリ
アプリケーションから呼ばれるGSS-APIを提供するライブラリ。デフォルト証明書獲得のための拡張GSS-APIもサポートする。セキュリティサーバ

2）ログインユーティリティ
ユーザと対話を行い、拡張GSS-APIを呼び出して証明書を獲得する証明書獲得フェーズを行う。また、クライアントにおいてGSS-APIライブラリから呼び出され、実際の認証プロトコルを処理する。

3）セキュリティデーモン
TGS、認証サーバの機能を実現する。

4）セキュリティクライアントデーモン
サーバにおいてGSS-APIライブラリから呼び出され、実際の認証プロトコルを処理する。

5）FTPクライアント／FTPサーバ
GSS-APIを呼び出し、ユーザ認証の機能を実現する。RFC959:FILE TRANSFER PROTOCOLに対するセキュリティ機能の拡張方法を提案しているInternet Draft : FTP Security Extensions [9] に準拠しユーザ認証を行う。

6）暗号ライブラリ
暗号方式に依存しない暗号APIを介して各種の暗号機能を提供する。

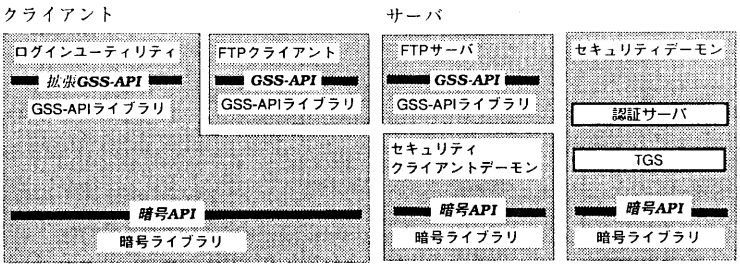

図4　プログラム構成

このような構成とすることで、「2．背景」で指摘した問題点を解決したユーザ認証システムを実現することができた。

6．　速度評価

試作したFTPを用いて処理速度の評価を行った。FTPのためのTCPコネクションを開始から、画面上にファイル一覧を表示するためのLISTコマンドへ終了までを、FTPコネクション接続時間(To)、コネクションを解放するためのコマンドを送ってから、実際のコネクション解放までを、FTPコネクション解放時間(Tc)として、RFC959、本システムの両方の認証方法における所用時間を測定した。測定はLANアナライザを用いて行い、実際のメッセージのやりとりされた時間から各時間を計算した。同じ測定を50回行い、その平均をとった。測定環境と結果を以下に示す。

使用機種

クライアント	ILIOS D433DV (i486DX/33MHz)
サーバ	HITACHI 3050RX/200(PA-RISC/33Hz)
セキュリティサーバ	HITACHI 3050RX/320(PA-RISC/50MHz)

認証方式	To（秒）	Tc（秒）
RFC959	1.60	0.07
本システム	3.36	0.52

FTPコネクション接続時間は約倍程度、FTPコネクション解放時間は7.4倍程度に増加した。しかし、絶対時間は3秒強、1秒弱と小さく、実際の操作においても違和感なく利用できた。

7．おわりに

本稿では、分散システムにおける従来のユーザ認証システムにおける問題点を指摘し、今回試作したユーザ認証システムの設計方針と実装について述べ、指摘した問題点が解決できること、さらに本システム上に試作したFTPにおいて、実用上問題の無い性能が得られることを明らかにした。

今後は、より安全なユーザ認証システムの実現を目指し公開鍵暗号方式や、さらにはPEMなどで採用されたX.509証明書を採用した認証プロトコルを実現していきたい。

謝辞

日頃、ご指導いただくNTTヒューマンインタフェース研究所第四プロジェクトチーム小柳津プロジェクトリーダ、有益なご助言いただいく、同研究所第四プロジェクトチームの皆様に感謝します。

参考文献

[1]Roger M. Needham and Michael D. Schroeder: "Using Encryption for Authentication in Large Networks of Computers.", CACM, Vol.12, No. 12, 993-999, 1978.

[2]ISO/IEC 9798-2, Information technology - Security techniques - Entity authentication - Part 2: Mechanisms using symmetric encipherment algorithms

[3]Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller: "Kerberos:An Authentication Service for Open Network System.", Proc.of USENIX 1988 Winter Conf.,191-202, USENIX, 1988

[4]Jhon T. Kohl:"The Evolution of theKerberos Authentication Service", Proc. of EurOpen 1991 Spring Conf.,EurOpen, 1991.

[5]S.M. Bellovin, M. Merritt, "Limitations of Kerberos Authentication System", ACM SIGCOMM Computer Communication Review, October 1990.

[6]宮口、栗原、太田、森田：" FEAL暗号の拡張" 、NTT R&D, 39, No. 10, 1439-1450, 1990.

[7]J. Linn:"Generic Security Service Application Program Interface", RFC1508, September, 1993.

[8]J. Wray:"Generic Security Service API : C-bindings", RFC1509, September, 1993.

[9]S. J. Lunt:"FTP Security Extensions", Internet Draft, March, 1995.