広域ネットワークにおける認証システムの設計と開発

白崎 博生
大阪大学基礎工学部情報工学科
sirasaki@ics.es.osaka-u.ac.jp

山口 英
奈良先端科学技術大学院大学 情報科学研究科
suguru@is.aist-nara.ac.jp

宮原 秀夫
大阪大学情報工学科
miyahara@ics.osaka-u.ac.jp

Abstract

本論文では、インターネットでの使用に耐えうる認証システムである SPLICE/AS-II の技術的概要、および、その実装について述べる。本システムの特徴は、プロセスの集合であるコミュニティという概念が導入され、さらに、通信によって取り扱う情報の重要度によりユーザが使用する暗号系やプロトコルを選択できることである。本システムの認証機構には公開鍵暗号を用いたプロトコルを使用している。そして、認証後は秘密鍵暗号による暗号化通信機構を提供することができる。

1　はじめに

インターネットは誕生以来オープンな環境として各種の実験が試みられながら発展を続けてきた。特に近年はインターネットを商業的取引の媒体とする実験などが見うけられる。さらに、商業プロバイダがインターネットに接続されたことにより、今後ますますユーザの数は爆発的に増加することが予想される。このようなネットワーク環境のもとでは誰がどの計算機を使っているのかを特定することが不可能である。このような理由により現在ユーザの認証技術が切に必要となりつつある。そして、ネットワークを流れるデータの盗聴や改竄などの不正行為から通信を保護する機構の実現も期待されている。

そこで、筆者らのグループではインターネットのようなオープンな広域ネットワークでの使用に耐えうる認証システム (SPLICE/AS-II) の設計と開発を行った。本システムの特徴は、コミュニティ(Community)という概念が導入されたことである。これにより今までにない柔軟なアクセスコントロールを行うことが可能になる。ユーザプロセスとサーバプロセスの相互認証機能には、管理すべき鍵の個数を抑え、その氾濫を避けるために公開鍵暗号を用いている。そして、プロセス間相互認証を行った後に、暗号化通信機能を提供し、不正行為からデータを守ることが可能としている。暗号化通信には処理速度の問題から、秘密鍵暗号を用いて実現している。認証プロトコルは特定の公開鍵暗号方式に依存しないように設計されており、アプリケーションプログラマ、あるいはユーザが通信において取り扱う情報の重要度により、使用する暗号系を選択することができる。本システムのプロトコルは、Needham と Schroeder のプロトコル [4] を基にしているが、コミュニティを導入するために大きな拡張を加えたものを使用している。

SPLICE/AS-II: An Authentication System for wide area network.
Hiroo SHIRASAKI
Faculty of Engineering, Osaka University
Suguru YAMAGUCHI
Nara Institute of Science and Technology
Hideo MIYAHARA
Faculty of Engineering, Osaka University

2　SPLICE/AS-II

図 1に SPLICE/AS-II におけるモデルを示す。SPLICE/AS-II システムはサーバ (Server)、クライアント (Client)、鍵配布サーバ (Key Distribution Server、以下 KDS と略す) の 3 種類の要素から構成される。そして、それぞれの構成要素はコミュニティという集合に属し、コミュニティはリージョンと呼ぶ部分集合に分けられる。

本システムで提供する認証機構は、公開鍵暗号を用いて実現し、認証後は秘密鍵暗号を用いた暗号化通信機構を提供する。

この章では SPLICE/AS-II の概要を説明し、第 3

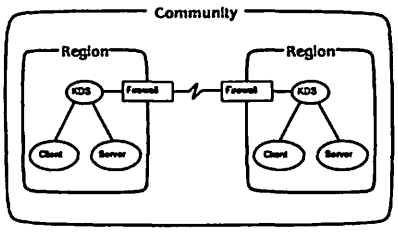

図 1: SPLICE/AS-II のモデル

節でプロトコルの精細な説明を行う。

2.1 コミュニティとリージョン

我々は本システムにコミュニティ(community) という概念を導入した。通常の社会では、人間は社会集団というグループを作る。そこには仕事関係のグループ、友人関係のグループ、ショッピング会員のグループ、レンタルショップ会員のグループなど多種多様なグループが存在する。そして、個人は複数のグループに属し、それぞれのグループにおける「立場」に応じてサービスを提供あるいは利用している。クレジットカードでの買い物にたとえてみる。ある個人はカード会員になることを申請し、カードを受け取ることによりグループに属することになる。複数のクレジットカードを持てば複数のグループに属することになる。クレジットカードのグループに属したカードユーザは、店での商品の購入時に、その店が扱っているカードに応じてそれを使い分けて代金を支払うということを行っている。また、店の方も複数のカード会社と提携することにより、複数のグループに対してサービスを提供している。

SPLICE/AS-II におけるコミュニティとは、この社会集団のグループをマッピングしたものである。ユーザは複数のコミュニティに属することができ、そのコミュニティ内でクライアントプログラムを使用することにより、コミュニティ内のサーバプログラムからサービスを受けることができる。一方、サーバも複数のコミュニティに属することができ、それぞれのコミュニティに属すクライアントからのサービス要求に答えることができる。

SPLICE/AS-II におけるクライアントとは、クライアントプログラムを起動したユーザとそのプロセスを指し、サーバとはサーバプロセスのことを指す。そして、これら 2 者に対して鍵を配布するプロセスである鍵配布サーバが加わり、これら 3 種類のプロセスがコミュニティの構成要素になる。

コミュニティを構成するメンバ(クライアントプロセス、サーバプロセス、鍵配布サーバ) はネットワーク上のどこに存在していてもかまわない。そして、いくつ存在してもかまわない。つまり、コミュニティのネットワーク的な広さや、コミュニティの大きさには制限はない。

しかし、鍵配布サーバの管理上、または、クライアント・サーバプロセスからのネットワーク到達性等の理由により、コミュニティをいくつかの部分集合に分けなければならない場合がある。SPLICE/AS-II では、この部分集合をリージョン (region) と名付けている。例えば、クライアントとサーバが同じネットワーク上にあり、鍵配布サーバがネットワーク的に遠いところに位置しているとする。このとき、クライアントと鍵配布サーバ間、そしてサーバと鍵配布サーバ間の通信に時間がかかるばかりか、ネットワーク資源の無駄使いにもなりかねない。ネットワーク的に近いところにクライアントとサーバが複数存在するならば、この部分をリージョンとしてまとめ、そのリージョン内に新たに鍵配布サーバを設置すれば応答性もよくなり、ネットワークを無駄に消費することを防ぐことができる。また、クライアントとサーバが位置するネットワークが防火壁 (firewall) によりインターネットと隔絶されている場合は、システムの実装上の理由によりそのネットワークをリージョンとして分割しなければならない。

2.2 認証方式

我々は SPLICE/AS-II で提供する認証について次の 3 種類を考えた。クライアントがサーバに対して認証を行うとき、データが正しい相手に伝わるということが保証できる。一方、サーバがクライアントに対して認証を行えば、データの発信元を保証することができる。この 2 つの認証を片方認証 (one-way authentication) と呼ぶ。そして、これら 2 つの認証を同時に行ったものを相互認証 (mutual authentication) と呼ぶ。

本システムではこれら 3 種類の認証方法 (サーバ認証、クライアント認証、相互認証) を提供し、ユーザ

がこれらを選べるように設計している。

2.3 暗号方式

秘密鍵暗号法においては、特定の2者が暗号化通信を行うには事前に秘密鍵を共有する必要がある。しかし、この鍵を第3者に知られることなく共有することは一般に困難である。一方、公開鍵暗号法においては、公開鍵と秘密鍵という2種類の鍵を用いてメッセージの暗号化・復号化を行う。そして、この公開鍵は第3者の手に渡っても問題はない。それは秘密鍵を持っている本人のみが暗号文の復号化を行えるからである。公開鍵暗号法の特徴としては、秘密鍵暗号に比べて鍵の配布が容易であるという長所がある。しかし、秘密鍵暗号に比べると処理速度が非常に遅いという短所もある。

本システムでは、公開鍵暗号法による電子署名の有効性を確かめることにより、それぞれのメンバが認証を行う。さらに、認証過程で2者間で秘密鍵暗号の鍵の共有を行う。そして、認証を行った後は、秘密鍵暗号法を用いた暗号化したメッセージを送受信する暗号化通信を実現している。

CPUの処理速度は年々速くなり、なおかつ価格も下がりつつある。つまり、高性能な計算機が安価で手に入るようになっているのである。計算機の処理速度の高速化は、暗号においては脅威であると言える。これは、鍵の全数探索[3]による暗号解読が容易になるからである。このことから、少々遅くても安全な通信を行いたい、あるいは、少々安全性は劣っても高速な通信を行いたいという要望があることが想定できる。

以上2つの理由により、特定の暗号系には依存しないような認証プロトコルと暗号化通信を設計している。なおかつ、それに用いる暗号の鍵の大きさは固定長ではなく、可変長が扱えるような実装を行った。つまり、ユーザは通信の重要度に応じて使用する暗号法やプロトコル、鍵の大きさを選択できるのである。現在は公開鍵暗号にはRSA[5][6]を、秘密鍵暗号にはDES[1]のみを実装しているが、今後、他の暗号系の実装を行う予定である。

2.4 ファイアウォール

人と人との関係はネットワークの形態に依存するものではない。しかし、近年ファイアウォール上でネットワークを流れるパケットのアクセス制御を行い、組織外ネットワークと内部ネットワーク間で自由な通信を制限しているサイトが数多く存在する。このようなファイアウォールにまたがったコミュニティを構成する場合、そのサイト内のメンバが孤立してしまう。そして、コミュニティ内で提供されているサービスを受けることができないのである。

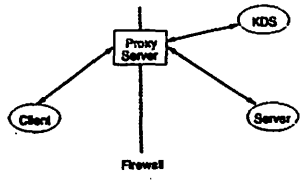

図2: SPLICE/AS-IIに対応したproxy server

そこで本システムでは、ファイアウォール上でSPLICE/AS-IIに対応した代理サーバ(proxy server)を設置することで解決することにした(図2)。しかし、代理サーバの設置は、それ自身がセキュリティホールになる恐れがある。そこで、代理サーバに全ての通信を転送させるようなことはさせず、アクセス制御機能を設けることにする。アクセス制御はコミュニティ名、リージョン名をもとにして行う。そして、アクセスが許可された場合、その方向は片方向・双方向のいずれかが設定できる。

2.5 名前

SPLICE/AS-IIを利用する全てのクライアント、サーバは一意な識別子(ID)を持つ。IDは名前、リージョン名、コミュニティ名の3つ組で構成される。

ID = { 名前, リージョン名, コミュニティ名 }

2.5.1 クライアントの「名前」

コミュニティに属するユーザは、リージョン内で一意な名前を持つ。コミュニティ内で同じ名前を持つユーザが複数存在しても、異なるリージョンに属しているのならば問題はない。また、複数のコミュニティに属しているユーザは、それぞれのコミュニティ毎に異なる名前を使用してもよい。もちろん、同じ名前を使用してもかまわない。リージョンの管理者が注意しなければならないのは、リージョン内に同名のユーザが存在しないように名前を与えることである。

ユーザが本システムに対応したプログラムを起動すると、そのプログラムはユーザのIDをそのまま引き継いで、クライアントプロセスとなる。

2.5.2 サーバの「名前」

サーバには次のような規則で名前付を行う。

Name = { サービス名@ホスト名 }

アプリケーションプログラマは、一つの計算機内で同じサービス名のプロセスが存在しないように名前をつける。ホスト名には、Domain Name Service(DNS)における正式名称 (cannonical name)、つまり、ホスト名にドメイン名を加えた名前を使用する。しかし、インターネットの中にはファイアウォールを用いて内部ネットワークと外部ネットワークを分断し、サイト内のホストの名前を外部に公開していないサイトが存在する。そのような場合は、代理サーバがホスト名からIPアドレスに変換できるようなホスト名を使用する。

以上の規則で全てのサーバに一意な名前をつけることができる。

複数のコミュニティに属するサーバにも同じ規則で名前付を行う。

以上の規則を用いれば、一つのIDから一意なサーバを求めることができる。

2.6 鍵配布サーバ (KDS)

鍵配布サーバはリージョン内の全てのメンバのID情報、公開鍵、秘密鍵、パスワードなど全ての情報を管理する。ゆえに、鍵配布サーバはネットワークセキュリティ上安全な計算機で実行させることを想定している。

鍵配布サーバの役割は、クライアントやサーバに鍵を配布することである。鍵配布サーバ自身も秘密鍵と公開鍵を持ち、それを用いて安全な鍵配布を行う。例えば、公開鍵配布の要求があったとき、自分の秘密鍵を用いて電子署名を行い、それとともに公開鍵を配布する。転送中の鍵の改竄防止や偽の鍵を受け取らないことを保証するためである。

メンバのアカウント作成も鍵配布サーバの役割である。鍵配布サーバが管理するデータベースに新規のアカウントを作成する際に、フロッピーディスクやIC Card、フラッシュメモリカードなどのリムーバブルな記録媒体にメンバのID情報、公開鍵、秘密鍵、パスワード、そして、鍵配布サーバの公開鍵を書き込む。以後ユーザはその記録媒体を用いて本システムを使用する。現在はフロッピーディスクドライブを持つ計算機の多さから、フロッピーディスクを使う実装を行った。しかし、計算機によってはフロッピーディスクドライブを持たないものが存在する。このようなときには、ユーザは鍵配布サーバに全ての情報の転送を要求しなければならない。この要求に答えるのも鍵配布サーバの役割である。

ユーザがフロッピーディスクを紛失、あるいは盗難にあったときはデータを第3者に読まれる恐れがある。そこで、フロッピーディスク内のデータは、全て暗号化して書き込む必要がある。そのときの暗号法は秘密鍵暗号法を用い、ユーザのパスワードを秘密鍵として暗号化・復号化を行う。

2.7 ユーザの移動

移動ホストの利用などの理由によりユーザが一時的に他のネットワークに移動する場合がある。このような場合にも本システムが使用できるための条件は、(1) 訪問先にSPLICE/AS-IIに対応化されたプログラムが用意してあること (2) 自分が属するリージョンの鍵配布サーバにコネクションを張れることの2点である。このようなときはネットワークの遅延時間の分だけレスポンスは悪くなるが、問題なく使うことはできる。

しかし、訪問先のネットワークと自分のネットワークの間にファイアウォールがあり、自分のネットワークとの間にコネクションを張ることが制限されている場合、鍵配布サーバにアクセスできなくなる。

そこで、訪問先の計算機から直接コネクションを張れる鍵配布サーバを一時的に、自分の鍵配布サーバとして利用し、鍵配布サーバはユーザが要求した鍵を転送することでこの問題は解決できる。

例えば、図3のClientは自分のリージョンの鍵配布サーバ KDS_1 にはアクセスできない。このとき、Clientが訪れたネットワークに鍵配布サーバ KDS_2 が存在するときは、Clientは以後 KDS_2 に鍵の配送を依頼する。Serverにアクセスするときは、まず KDS_2 に依頼し、KDS_2 が KDS_1 にServerの公開鍵を要求する。最後に KDS_2 がClientにそれを転送するのである。

記号	意味
KDS	鍵配布サーバ
S	サーバ
C	クライアント
L	クライアントのログイン名
Com	コミュニティ名
ID	識別子
PW	KDS に登録されている暗号化されたパスワード
PW'	ユーザが入力する暗号化されたパスワード
U	ユーザ
SK_X	X の秘密鍵
PK_X	X の公開鍵
SSK	セッション鍵。十分大きな整数 (乱数)
$DS(M)^{SK}$	秘密鍵 SK を用いたメッセージ M の電子署名
$\{M\}^{PK}$	PK を鍵としてメッセージ M の公開鍵暗号を用いた暗号化
$\{M\}^{SK}$	メッセージ M と電子署名 $DS(M)^{SK}$ に展開
$[M]^{K}$	K を鍵としてメッセージ M の秘密鍵暗号を用いた暗号化

表 1: プロトコルの表記に用いる略称

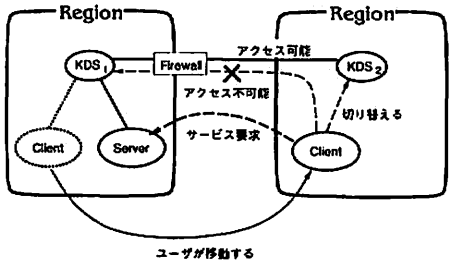

図 3: ユーザが移動する場合

3 プロトコル

この章では、鍵配布と認証機構を実現するために設計したプロトコルについて説明する。プロトコルの表記については表 1を用いる。

プロトコルの説明を行う前に、2 つの前提条件を用意する。(1) クライアントは、コミュニティ名を元にしてリージョン内の鍵配布サーバのホスト名を知ることができる。(2) 各鍵配布サーバは、サーバの ID をもとにしてそれを管理する鍵配布サーバのホスト名を知ることができる。

3.1 初期化

ユーザがシステムを使用する際、ユーザは鍵配布サーバの公開鍵、自分の秘密鍵と公開鍵を持っていなければならない。通常ユーザはこれらをフロッピーディスク内に保存し、計算機本体に付属するフロッピーディスクドライブから読み込む。さらに、ドライブのない計算機のためにこれらの情報を転送するプロトコルを実装する必要がある。その他にこのプロトコルを使用するのは、鍵配布サーバの鍵が更新されたときが挙げられる。

このプロトコルで注意しなければならない点は、ユーザが偽の鍵を受け取らないようにすることである。この問題を解決するためにパスワードを用いることにした。

a. ユーザは初期化プログラムを起動し、コミュニティ名とユーザ名を入力する。

$$U \rightarrow C : L, Com$$

b. クライアントは ID_C とセッション鍵を鍵配布サーバに送る。セッション鍵には十分大きな乱数を用いる。初期化のプロトコルではこの鍵を replay attack を検出するために用いている。各計算機のタイムスタンプ [2] を用いても同様のことができるが、コミュニティ内の計算機全てのクロックを同期させる必要がある。これを仮定することは現実では難しいので、セッション鍵として乱数を用いることにした。

$$C \rightarrow KDS : ID_C, SSK_1$$

c. 鍵配布サーバは、鍵配布サーバの公開鍵をデータベースから取りだし、ユーザのパスワードで暗号化する。次に、ID_{KDS} と ID_C、SSK の電子署名を作成

し、これらをクライアントに送信する。

$KDS \rightarrow C$: $[PK_{KDS}]^{PW}$, { ID_{KDS}, ID_C, SSK_1 }$^{SK_{KDS}}$

d. クライアントプログラムはユーザにパスワード入力を要求する。そのパスワードを用いて、まず$[PK_{KDS}]^{PW}$から鍵配布サーバの公開鍵を取り出す。つぎに、ID_{KDS}とID_C、SSKの電子署名の有効性が確かめられれば、この公開鍵は本物であると判断する。

$U \rightarrow C : PW'$

鍵配布サーバの公開鍵 (PK_{KDS}) のみが必要なときはここで処理は終了する。さらに、自分の秘密鍵と公開鍵が必要な場合は次の処理に進む。

e. クライアントは新たにSSK_2を生成し、ID_CとID_{KDS}、SSK_2をPK_{KDS}で暗号化し鍵配布サーバに送信する。ゆえに、SSK_2は鍵配布サーバのみが知ることができ、第3者は知ることができない。

$C \rightarrow KDS$: ID_C, ID_{KDS}, { ID_C, ID_{KDS}, SSK_2 }$^{PK_{KDS}}$

f. 鍵配布サーバはクライアントの秘密鍵と公開鍵をデータベースから取りだし、クライアントのパスワードで暗号化する。それをさらにSSK_2で暗号化してクライアントに送信する。SSK_2を知っているのは鍵配布サーバだけであるので、このメッセージがうまく復号化できれば、攻撃者によりメッセージが改竄されなかったと判断できる。

$KDS \rightarrow C$: ID_{KDS}, ID_C, [ID_{KDS}, ID_C, [SK_C, PK_C]^{PW}]^{SSK_2}

3.2 鍵配布

3.2.1 公開鍵の配布

クライアントがサーバの公開鍵を要求するときのプロトコルを説明する (図4)。ここでは、クライアントは鍵配布サーバ1の公開鍵を持っている。鍵配布サーバ1は鍵配布サーバ2の公開鍵を持っている。以上の2つを前提条件とする。

なお、以下の説明の「サーバ」を「クライアント」に置き換えると、サーバがクライアントの公開鍵を要求するときのプロトコルになる。

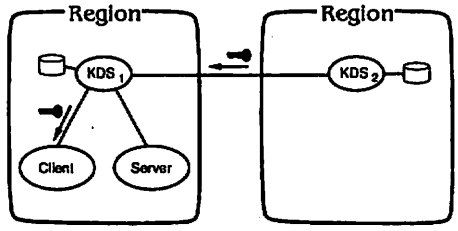

図 4: クライアントがサーバの公開鍵を要求

A) 同一リージョン内のサーバの公開鍵を要求するとき

a. クライアントはサーバのIDであるID_Sとセッション鍵を鍵配布サーバに送る (プロトコル 1.1)。

$C \rightarrow KDS : ID_C, ID_{KDS}, ID_S, SSK$ (1.1)

b. 鍵配布サーバはID_S内のリージョン名を調べ、自分の管理するリージョンならば、データベースからサーバの公開鍵を取りだしクライアントに送る。このとき、クライアントに「確かに鍵配布サーバからの送信である」ことを確信させるために、鍵配布サーバは公開鍵とそれに対する電子署名を同時に送信する。このときクライアントに送信する内容 (プロトコル 1.2) は、公開情報のみであるので、暗号化する必要はない。

$KDS \rightarrow C$: { $ID_{KDS}, ID_C, ID_S, PK_S, SSK$ }$^{SK_{KDS}}$ (1.2)

c. クライアントは、受け取ったデータを鍵配布サーバの公開鍵を用いて電子署名の有効性を確かめる。また、セッション鍵の同一性を調べ、もし一致しなければ replay attack が行われたと判断する。

B) 他リージョンのサーバの公開鍵を要求するとき

d. もし、(b) の時点で他リージョンのサーバと判断できたならば、鍵配布サーバ1は、ID_Sを管理する鍵配布サーバ2に公開鍵を要求する (プロトコル 1.3)。

$KDS_1 \rightarrow KDS_2$: $ID_{KDS_1}, ID_{KDS_2}, ID_S, SSK_{KDS}$ (1.3)

e. 鍵配布サーバ2は、サーバの公開鍵を鍵配布サーバ1に送る (プロトコル 1.4)。

$KDS_2 \rightarrow KDS_1$: { ID_{KDS_2}, ID_{KDS_1}, ID_S, PK_S, SSK_{KDS} }$^{SK_{KDS_2}}$ (1.4)

f. サーバの公開鍵を受け取った鍵配布サーバ1は、クライアントにそれを転送する (プロトコル 1.2)。

3.3 認証

本システムでは、「サーバ認証」「クライアント認証」「相互認証」を行う3種類のプロトコルを設計した。前の2つを片方認証、最後のものを相互認証という。認証プロトコルでは公開鍵暗号を多用するので、どうしても速度が遅くなってしまう。そこで、ユーザあるいはアプリケーションプログラマが通信の重要度により適切なプロトコルを選ぶことができるようにした。認証プロトコルの後はSSKを秘密鍵として暗号化通信を行う。

A) クライアントがサーバを認証

$C \rightarrow S : ID_C, ID_S, \{\ ID_C, ID_S, SSK\ \}^{PK_S}$
$S \rightarrow C : ID_S, ID_C, \{\{\ ID_S, ID_C, SSK,\ \}^{SK_S}\}^{PK_C}$

B) サーバがクライアントを認証

$C \rightarrow S : ID_C, ID_S, \{\{\ ID_C, ID_S, SSK\ \}^{SK_C}\}^{PK_S}$
$S \rightarrow C : ID_S, ID_C, \{\ ID_S, ID_C, SSK,\ \}^{PK_C}$

C) 相互に認証

$C \rightarrow S : ID_C, ID_S, \{\{\ ID_C, ID_S, SSK_C\ \}^{SK_C}\}^{PK_S}$
$S \rightarrow C : ID_S, ID_C, \{\{\ ID_S, ID_C, SSK_S, SSK_C\ \}^{SK_S}\}^{PK_C}$
$C \rightarrow S : ID_C, ID_S, \{\ ID_C, ID_S, SSK_S\ \}^{PK_S}$

3.4 コネクションの確立

クライアントがサーバとコネクションを確立するまでには4つのフェーズがある。これらを始めから順番に次のように名付けた。コネクション要求フェーズ (Connection Request Phase)、交渉フェーズ (Negotiation Phase)、公開鍵転送フェーズ (Get Public Key Phase)、認証フェーズ (Authentication Phase)。この節ではクライアントがコネクションを確立するまでの手順を説明する。

3.4.1 コネクション要求フェーズ

図5の例では、クライアントとサーバ間に proxy server が一つだけあるときの例を示した。proxy server がこのフェーズのパケット (CR packet) を受け取ると、最初にアクセス制御のチェックを行う。これが許

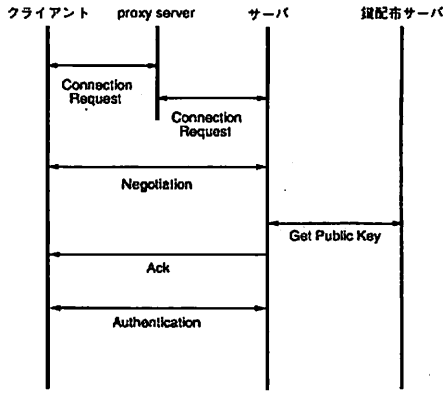

図 5: コネクションが確立するまで

可されるならばパケット転送用のプロセスを生成する。もし、拒否の場合はただちにコネクションを切断する。CR packet にはクライアントの ID が含まれており、proxy server とサーバはこの段階でログを出力する。

3.4.2 交渉フェーズ

このフェーズでは、以後の認証フェーズで使用するプロトコル、そして暗号化通信で使用する暗号系の種類をクライアントとサーバが交渉する。交渉プロセスは、まず最初にクライアントが希望するプロトコルと暗号系をサーバにリクエストする。もし、サーバがそれらを受け入れるならば Ack パケットをクライアントに送りこのフェーズを終了する。サーバがそれらのうち片方、あるいは両方を拒否するならば Nack パケットともにサーバの希望するプロトコル、または暗号系をクライアントにリクエストする。クライアントがそれらを受け入れるならば Ack パケットをサーバに送りこのフェーズを終了する。受け入れられないならば交渉決裂になり、コネクションを切断する。

3.4.3 公開鍵転送フェーズ

使用するプロトコルが決まれば、そのプロトコルで用いる公開鍵が必要になる。次に行うのは、サーバが鍵配布サーバからクライアントの公開鍵を取得するフェーズである。エラーなく受け取ることができれば、サーバはクライアントに Ack パケットを送る。そ

うでなければNackパケットを送りコネクションを切断する。

3.4.4 認証フェーズ

このフェーズでは3.3で説明したプロトコルを用いて認証を行う。これに成功すれば、以後クライアントとサーバは暗号化通信を行う。失敗すればコネクションを切断する。

4 実装とアプリケーションへの適用

我々はOSに依存しないシステムを作ることを目標としたので、全てのプログラムをアプリケーションレイヤで実装した。システムのパッケージには鍵配布サーバ、リンク用ライブラリ、システム管理ツールが含まれている。telnetやftpなどの既存のプログラムをSPLICE/AS-IIに対応化させるためには、プログラムを若干ではあるが修正し、リンク用ライブラリをリンクする必要がある。

残念であるが全てのプログラムをSPLICE/AS-II対応化させることはできない。本システムはコネクションの確立時に、そのコネクションに対して認証を行う。よって、コネクションの確立・切断を頻繁に繰り返すプログラムは速度の問題から実用にならないものになることが予想される。また、本システムはTCPコネクションを使用することを前提にしているので、UDPを使用するプログラムは対応化できない。最後に、本システムはコミュニティに属するユーザにのみサービスを提供することを前提として設計しているので、Anonymous FTPなどの不特定多数のユーザに対してサービスを行うプログラムには本システムは不適切である。

本システムに適しているプログラムは、特定の人だけにサービスを提供したいようなものである。例えば、データベース検索、ファイルサーバ、メッセージ転送システムなどが挙げられる。

5 まとめ

本論文では、インターネットでの使用を想定した認証システムであるSPLICE/AS-IIの設計と開発における技術的概要と、その実装について述べた。SPLICE/AS-IIでは他の認証システムにはないコミュニティという概念を導入し、柔軟なアクセス制御機能を提供することができる。また、コミュニティをリージョンに分け、ファイアウォールで切断されているサイトにはproxy serverを設置することにより、広域分散ネットワークでの認証サービス提供することを可能とした。

システムは特定の暗号系には依存しないように、そしてユーザが鍵の大きさを選択できるように設計した。このことにより、安全性よりも速度を重視するユーザや、速度よりも安全性を重視するユーザの要望にも答えることができる柔軟なサービスが提供できる。

現在SPLICE/AS-IIでは実装を行ってないところが何箇所かあるが、今後も実装作業を続け、その後さまざまなツールのSPLICE/AS-II対応化を行っていく。

謝辞

最後に、さまざまな面で御協力いただきました大阪大学の宮原研究室の方々、奈良先端科学技術大学院大学の山本研究室の方々と、WIDE Projectの皆様に心から感謝致します。

参考文献

[1] Data encryption standard. Fips pub46, National Bureau of Standards, Washington, D.C, 1977.

[2] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols.

[3] W. Diffe and M. Hellman. Exhaustive cryptanalysis of the nbs data encryption standard. *Computer*, 10(6):74–84, June 1977.

[4] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers.

[5] A. Shamir R. Rivest and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems.

[6] Philip Zimmermann. A proposed standard format for rsa cryptosystems. *IEEE Computer*, pages 21–34, September 1986.