マルチメディア通信と分散処理 51－8
（1991. 7. 5）

大規模広域ネットワークにおけるセキュリティ機能の実現

山口 英　　岡山 聖彦　　宮原 秀夫

大阪大学基礎工学部情報工学科

あらまし: 広域ネットワーク環境においては、運用を行なっていく上でセキュリティの問題を早急に解決する必要がある。WIDE Project では、WIDE Internet でのセキュリティの基礎となる利用者認証システム SPLICE を大阪大学が中心となって開発してきた。しかしながら、当初開発した SPLICE には運用上の幾つかの問題が指摘されている。そこで、それ羅の問題を開発するために現在の SPLICE での技術を基盤として新たな利用者認証システム SPLICE-II を開発している。SPLICE-II では、慣用系暗号を用いた利用者認証プロトコルを用い、運用面を重視した管理構造を導入している。本稿では SPLICE-II の設計と実装について述べる。

Design and Implementation of Security Functions for Large Scale Wide Area Networks

Suguru Yamaguchi　　Kiyohiko Okayama　　Hideo Miyahara

Dept. of Information and Computer Science,
Faculty of Engineering Science, Osaka University.

1-1 Machikaneyama, Toyonaka, Osaka 560 JAPAN.

Abstract: In a large scale distributed environment or large open networks like WIDE Internet, an authentication system is a fundamental building block for providing security mechanism. We have developed a trusted third-party authentication system called *SPLICE* for the WIDE Internet. However, in view of its operation in large open networks, its management structure and cipher should be improved. Therefore, we started to develop new authentication system called *SPLICE-II* based on our experiments in the operation of SPLICE. We introduced the authentication protocol based on a conventional encryptosystem which was originally proposed by Needham et al. Furthermore, its management scheme is adequate to operate it in large open networks. This paper describes the design of SPLICE-II, and its implementation.

1 はじめに

近年、コンピュータネットワークの発展によって様々な組織が広域ネットワークに接続され、メッセージ交換やファイル転送、仮想端末機能などを利用できる環境が一般的となってきた。さらに、大学や企業でも、組織内の LAN を相互接続し、規模の大きなネットワーク環境を構築し始めている。このような環境が広がるにつれて、ネットワークに接続された資源の不正利用やネットワーク上を流れる情報の不正な入手(盗聴) といった、いわゆるセキュリティの問題をどのように扱っていくかを解決することが緊急かつ重要な問題となってきた。特に、広域ネットワーク環境では、異なる管理母体に管理されたネットワークを相互接続しているために、セキュリティの問題をより早急に解決しなければならない。

WIDE Project では、1988 年より WIDE Internet と呼ばれる大規模広域ネットワークを基盤とした分散処理環境の構築を行ってきた [1]。現在の WIDE Internet では、東京、京都、大阪、福岡の各ネットワークオペレーションセンタ (NOC) を中心に大学、研究機関を結ぶ大規模なネットワークを形成している。WIDE Project では、WIDE Internet でのセキュリティの基礎となる利用者認証機構 SPLICE を大阪大学が中心となって開発を行ってきた [2, 3]。しかしながら、開発された SPLICE にはさまざまな問題が指摘されている。そのため、それらの問題を克服するために現在の SPLICE での技術を基盤として新たな利用者認証機構 SPLICE-II の開発を行っている。本稿では、SPLICE-II の概要、構成について述べる。

2 SPLICE での問題点

大阪大学で開発を行ってきた SPLICE は、大規模広域ネットワーク環境での運用を意識した利用者認証システムである。SPLICE では、Needham 等によって示された公開鍵暗号を用いた利用者認証プロコトコル [4] を基本にシステムを構成している。さらに、管理構造として Internet における DNS (Domain Name System)[5] での名前構造に対応した階層的なドメイン構造を導入し、ドメイン間の鍵の配送を階層的に行うことで広域ネットワークでの分散管理・運用を実現している (図 1参照)。SPLICE は、このような特徴を持ったシステムであるが、次のような問題点が指摘されている。

2.1 暗号方式と性能

現在の SPLICE での最大の問題点は、認証サーバの処理が遅いことにある。利用者認証機能を提供する SPLICE は、ネットワーク環境における基礎となる機能を提供するものであり、多くのクライアントからの要求に素早く反応できることが必要である。このため、どこがボトルネックになっているかを調べるために、SPLICE の認証サーバを UNIX の profile 機能を用いて調べたところ、暗号／復号処理部分の CPU 使用量がもっとも多いことが明かになった。

SPLICE では、Needham 等によって提案された暗号技術を基礎としている利用者認証プロトコルを使用している。このため、SPLICE ではさまざまなデータの暗号化・復号化の処理が非常に多い。したがって、SPLICE が実用的なサービスを提供するためには、SPLICE で用いている暗号方式が効率的であり、さらに、実装がうまく行われている必要がある。

SPLICE では、暗号方式として公開鍵暗号の RSA[6] を用いており、実装をソフトウェアで行なっている。しかしながら、RSA では暗号鍵として非常に大きな整数(SPLICE では 64byte) を用いることと、暗号化、復号化の段階で大量の剰余演算が必要であることから、ソフトウェアによる実装では非常に効率が悪い。また、ソフトウェアの実装のままで処理を高速化するには、鍵の長さを短くすればよいが、鍵の長さは暗号の安全性との間でトレードオフの関係があり、現在の 64byte 以下の長さにすることは安全性の面から考えて難しい。さらに、現状で公開鍵暗号を実用化しているものでは、その多くがハードウェアによる実装を行っていることから、SPLICE でも専用のハードウェアを導入して暗号関連の処理を根本的に

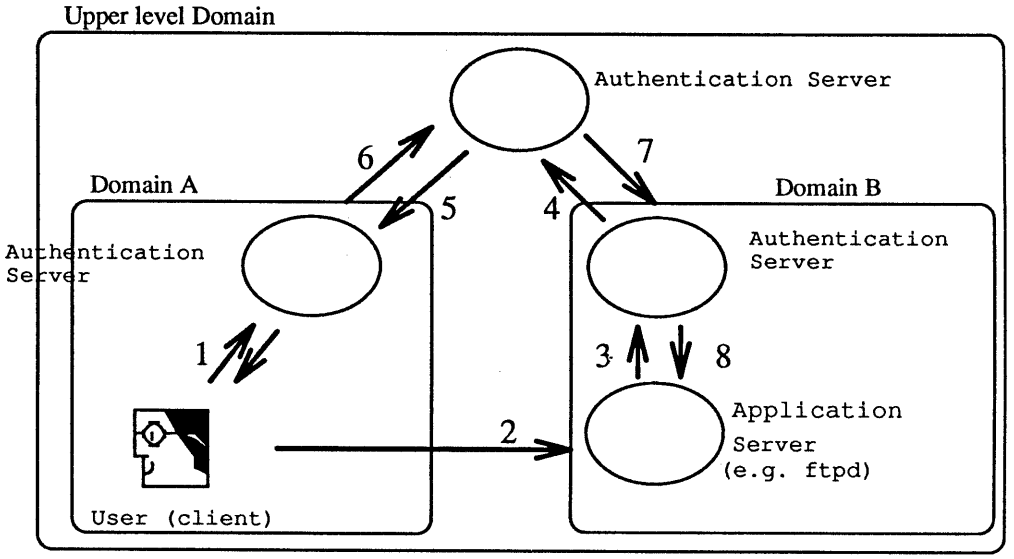

図 1: SPLICE での階層的な管理構造

高速にする実装も考えられるが、SPLICE は広域ネットワークに接続されたワークステーションなどでの運用を前提にしており、対象となる特別のハードウェアの付加を必要とする実装は運用面から考えて現実的ではない。このようなことから、何らかのより効率の良い暗号方式を導入する必要があると考えられる。

2.2 管理構造

現在の SPLICE では、DNS が用いているドメイン構造に対応した階層的なドメインによる管理構造を導入している。各ドメインには、各々一つの認証サーバを配置し、各サーバはそのドメインでの認証サービスを提供する。このため、認証サービスを維持するためには、各ドメインでは常に認証サーバが稼働していることが必要である。しかしながら、このような構成では次のような問題点がある。

(a). サーバがクラッシュした場合。この場合、サーバが復帰するまで利用者認証のサービスを受けられなくなる。利用者認証は、ネットワークサービスの根幹に関わるものであり、サーバがクラッシュした場合にサービスが停止することの影響は大きい。例えば、すべてのネットワークサービスが利用不可能になることが考えられる。

(b). SPLICE では、DNS の名前構造に対応したドメイン構造を持っている。一般的に各組織内では管理組織に応じてサブドメインを構成することが多いが、組織内のサブドメインを構成していない組織もある。また、サブドメインを構成していても、そのサブドメインに属する計算機の台数や利用者数が膨大な場合もしばしばある。このため、DNS に対応したドメイン構成では、認証サーバに対してトラヒックが集中し、処理が円滑に行えない場合が考えられる。

この問題点は、認証サーバの多重化を行なうことで解決することができる。しかしながら、多重化されたサーバ間では、登録されている利用者の情報に関して整合性がとれている必要がある。そのため、利用者情報データベースのサーバ間での安全な伝搬、更新を行なうための機構を新たに開発する必要がある。

2.3 認証サーバによって提供される情報の不足

利用者認証機構では、すべての認証サーバの管理・運用は安全に行われる必要がある。SPLICE では、各ドメインに認証サーバを配置することを考えている。この場合、すべてのドメインで均一なレベルで運用が行われるかどうかの保証はなく、あるドメインではサーバの運用がルーズに行われる可能性がある。

このようなことを考えると、認証システムの運用を行なっていく場合、特定の認証サーバからの認証情報を信用しないというような処理を行なう必要が出てくる可能性がある。このためには、利用者認証機構で提供される情報として、「通信相手がいったい誰なのか」という情報のほかに、「認証の情報を発行したサーバはどれなのか」という情報が付加されていることが望ましい。現在の SPLICE では、この情報が含まれておらず、認証プロトコルの変更が必要である。

3 SPLICE-II における認証モデル

前章で述べたように DNS の名前構造に対応した管理構造には様々な問題があることが明らかになった。このため、SPLICE-II では、より柔軟な認証モデルに基づいて利用者認証機構を構成する。

3.1 用語

以後の議論を明確にするために次のように用語を定義する。

リージョン (region): 一つの認証サーバが提供するサービスの提供範囲を表す。リージョンは、DNS の一つのドメインを越えては定義されない。例えば、DNS でのドメイン `ics.osaka-u.ac.jp` 内で運用される認証サーバでは、そのリージョンはドメイン `ics.osaka-u.ac.jp` の範囲を越えることはない。各認証サーバは、一つのコミュニティ(後述) に属している。

コミュニティ (community): コミュニティは認証サーバのグループを表す。このグループは、管理・運用方針に従って構成される。したがって、一つのコミュニティでは、一つまたは複数のドメインによって構成される。各コミュニティには、コミュニティ内でのリージョン間の認証を管理するコミュニティリーダ (community leader) がある。コミュニティリーダは、リージョン間で発生する認証での安全な鍵の配送を管理する。各コミュニティは名前 (コミュニティ名) を持ち、識別される。

利用者 (user): 利用者は少なくとも一つのコミュニティに属する。各利用者は、同じリージョンにある認証サーバに対して、認証に必要な情報を登録する。

認証サーバ (authentication server): ある一つのコミュニティに属し、同じコミュニティに属するサーバ、クライアントに対して認証サービスを提供するプロセス。

クライアント (client): ここでは、利用者が起動したプログラムを差し、サーバから提供されるサービスを利用者に提供するインタフェースのプロセス。クライアントはそれを起動した利用者のコミュニティに属していると考える。

サーバ (server): 何らかのネットワークサービスを提供するプロセス。サーバは、各ワークステーションの特定の利用者に属していると考える。UNIX ワークステーションでは一般的に root に属す。ここで、サーバは複数のコミュニティに属すことができる。

記号	意味
C	クライアント
S	サーバ
AS	認証サーバ
t	メッセージを発行したタイムスタンプ (time stamp)
l	メッセージの有効期間 (life time)
K_A	プロセス (利用者) A の暗号鍵
N_A	プロセス (利用者) A が属すコミュニティ
$\{M\}^K$	鍵 K によるメッセージ M の暗号化
A → B: *msg*	プロセス A からプロセス B へのメッセージ *msg* の送付

表 1: プロトコルの表記に用いる記法

3.2 暗号方式とプロトコル

SPLICE-II では、暗号方式として慣用系暗号を用いることにした。これは、ソフトウェアで暗号器を構成した場合でも、公開鍵暗号と比較して十分高速な処理速度を提供でき、さらに、多くのワークステーションで DES が利用できることによる。

認証プロトコルとしては、Needham 等が [4] で提案した方式を基礎として構成する。認証に必要なプロトコルとして、(1) 利用者自身の認証、(2) 同じリージョン内でのクライアント・サーバ間での認証、(3) 異なるリージョン間でのクライアント・サーバ間での認証が必要である。以下、それぞれについて述べる。なお、これ以後、表 1に示したような記法を用いて記述を行う。

3.2.1 名前空間

各利用者を識別するために、各利用者は次のような名前をつけ、それによって識別する。

username.community@host.DNS-domain

例えば、利用者 suguru がコミュニティ WIDE に属していており、ホスト `raicho.rd.ecip.osaka-u.ac.jp` からサービスを利用する場合、その利用者は

`suguru.WIDE@raicho.rd.ecip.osaka-u.ac.jp`

という名前を持つ。サーバの場合も同様の名前を用いる。例えば

`telnet.WIDE@raicho.rd.ecip.osaka-u.ac.jp`

は、同じホストでのコミュニティ WIDE に属する telnet サーバを表す。

3.2.2 利用者自身の認証

利用者自身の認証は、一般にワークステーションへのログイン時に行う。ここでは、通常のシステムへのログインでのパスワードの入力と同じように、認証サーバに登録されているパスワードの入力を利用者に行わせ、これが事前に登録されているパスワードと同じものかどうかをチェックすることで行う。これは、次のようなプロトコルに従って行う。ここで、利用者を A、同じリージョンの認証サーバを AS、input を入力されたパスワードとする。また、メッセージ中の A, AS はそれぞれの名前を表す。

$$\text{A} \rightarrow \text{AS}: \quad A, \{A,t\}^{input} \qquad (1a)$$
$$\text{AS} \rightarrow \text{A}: \quad \{A,AS,t,l\}^{K_{AS}} \qquad (1b)$$

(1a) では、入力されたパスワードを元にメッセージを構成し、認証サーバ側でメッセージを事前に登録されているパスワードで復号を試み、メッセージが正しく復号されるかどうかをチェックする。正しく復号された場合は (1b) のメッセージをクライアント側に送り返す。ここで、送り返されてきた

$$\{A,AS,t,l\}^{K_{AS}}$$

については、これ以後の認証サーバへアクセスする場合に必ず付加されるものであり、利用者側ではこれを保存しておく。以後の説明では、これを T という記号を用いて表す。

3.2.3 同一リージョン内のクライアント・サーバ間の認証

同一リージョン内のクライアント・サーバ間の認証のプロトコルは以下のようになる。ここで、CK は通信用に認証サーバが生成した一時的な暗号鍵であり、乱数よりランダムに生成される。id も同様に乱数により生成される整数である。これにより、相互の認証を行うことができる。

$$\text{C} \rightarrow \text{AS}: \quad \text{T,S} \qquad (2a)$$
$$\text{AS} \rightarrow \text{C}: \quad \{C,S,AS,CK,t\}^{K_S},\{CK\}^{K_C} \qquad (2b)$$
$$\text{C} \rightarrow \text{S}: \quad \text{C,S},\{C,S,AS,CK,t\}^{K_S},\{id\}^{CK} \qquad (2c)$$
$$\text{S} \rightarrow \text{C}: \quad \text{S,C},\{id+1\}^{CK} \qquad (2d)$$

3.2.4 異なるリージョン間での認証

異なるリージョン間で認証が必要になる場合を考える。ここで、クライアント C がローカルのリージョンの認証サーバに、異なるリージョンのサーバ S へのメッセージの生成を要求した場合、前節で示したプロトコルによれば、

$$\{C,S,AS,CK,t\}^{K_S}$$

の部分については、ローカルの認証サーバは生成することはできない。これは、サーバ S の鍵については異なるリージョンの認証サーバが管理を行っているからである。

この問題を解決するために、各コミュニティではコミュニティリーダと呼ばれるサーバを用意する。コミュニティリーダは同じコミュニティに属する認証サーバとの間で安全なメッセージの交換を保証する。このため、コミュニティサーバに対しては、各認証サーバが事前に通信用の鍵を登録しておく。この鍵を用いてメッセージを暗号化し、コミュニティサーバをメッセージの仲介者とすることで、安全に通信を行うことができる。そこで、異なるリージョンのサーバに対して認証が必要な場合、クライアントから要求を受けた認証サーバ X は、他の認証サーバ Y に対して、次のような通信を行う。ここで、この通信はコミュニティサーバ CS を経由しており、安全に通信できるものとする。

$$\text{X} \rightarrow \text{CS}: \quad \{C,S,X,CK\}^{K_X} \qquad (3a)$$
$$\text{CS} \rightarrow \text{Y}: \quad \{C,S,X,CK\}^{K_Y} \qquad (3b)$$
$$\text{Y} \rightarrow \text{CS}: \quad \{\{C,S,X,CK,t\}^{K_S}\}^{K_Y} \qquad (3c)$$
$$\text{CS} \rightarrow \text{X}: \quad \{\{C,S,X,CK,t\}^{K_S}\}^{K_X} \qquad (3d)$$

これによってサーバ S の鍵に依存した部分をクライアントは入手することができる。これにより、クライアントはメッセージをサーバに送り、認証を行う。

4 SPLICE-II の構成

SPLICE-II は、次のものから構成される。

認証サーバ: 一つのリージョンには、複数の認証サーバを用意することができる。このとき、一つのサーバが主サーバ (primary server)、それ以外のサーバが副サーバ (secondary server) として扱われる。これは、先に述べたサーバがクラッシュした場合にもサービスを続けるために用意される。副サーバは、起動すると主サーバから利用者の情報を登録したデータベースを転送する。この時、主サーバ側では副サーバのリストを用意しており、それ以外のホストからの要求を無条件に受け付けないようにする。また、転送中のデータは全て暗号化されており、盗聴を防止する。

クライアント用ライブラリ: 利用者認証サービスを利用するためのライブラリで、認証サーバ、および、クライアント・サーバ間での通信のためのスタブが用意されている。

暗号ライブラリ: SPLICE-II では、DES をソフトウェアで実現したものを使用している。これをユーザプログラムでも利用できるようにライブラリとして提供してる。例えば、通信データの暗号化などに利用することができる。

管理用ユーティリティ: 管理用ユーティリティとしては、認証サービスを利用するときに最初に起動する asinit と、認証サーバのデータベースに登録してあるデータを利用者が更新するための asdb という二つのコマンドが用意されている。

5 おわりに

現在、SPLICE-II の実装が進められており、運用試験を行っている段階である。さらに、今年中には WIDE Internet での本格的な運用を目標としている。今後の課題として、TELNET, FTP などの一般的なネットワークサービスに組み込み、その有効性を実証することが挙げられる。また、ここで用いている認証プロトコルの安全性の検証や改良などを行う必要があると考えらえる。

参考文献

[1] Jun Murai, Akira Kato, Hiroyuki Kusumoto, Suguru Yamaguchi, and Tomomitsu Sato. Construction of a Widely Integrated Distributed Environment. In *Proceedings of IEEE Region 10 Conference on Computer and Communication System*, pages 21.3.1–21.3.4, 1989.

[2] Suguru Yamaguchi and Hidetoshi Unno. Security Issues in WIDE project. In *Proceedings of the 4th Joint Workshop on Computer Communications*, pages 141–149, 1989.

[3] Suguru Yamaguchi, Kiyohiko Okayama, and Hideo Miyahara. Design and implementation of an authentication system in WIDE Internet environment. In *Proceedings of IEEE Region 10 International Conference TENCON'90*, pages 653–657, 1990.

[4] Roger M. Needham and Michael D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Communications of the ACM*, 21, No. 12:993–999, 1978.

[5] P. V. Mockapetris and K.J. Dunlap. Development Of The Domain Name System. In *Proceedings of ACM SIGCOMM'88*, 1988.

[6] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21, No. 2:120–126, 1978.