社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

セッション毎に IPv6 アドレスを変動させる安全な通信方式とそのアドレス更新手法

西田　和生† 　阿多　信吾†† 　北村　浩††† 　村田　正幸†

† 大阪大学大学院情報科学研究科 〒 565-0871 大阪府吹田市山田丘 1-5
†† 大阪市立大学大学院工学研究科 〒 558-8585 大阪府大阪市住吉区杉本 3-3-138
††† 日本電気株式会社／電気通信大学 〒 108-8557 東京都港区芝浦 2-11-5
E-mail: †{k-nisida,murata}@ist.osaka-u.ac.jp, ††ata@info.eng.osaka-cu.ac.jp, †††kitamura@da.jp.nec.com

あらまし　本稿では、外部ネットワークを経由した通信をより安全に行うために、IP アドレスを通信セッションの「鍵」として使用し、セッション毎に変動させる新しい通信手法を提案する。セッション毎に異なる鍵（IP アドレス）を用いることで、より、外部からの通信をより安全に実現する。本稿では、上記通信手法を実現するための主要課題であるサーバ待ち受けモデル、アドレス更新および同期処理について考察し、さらに実装モデルを示すことにより実現可能性についても明らかにする。

キーワード　鍵アドレス, 待ち受けモデル, IPv6, セキュリティ, Unified Multiplex, リモートアクセス

A Secure Communication Style to Use Changable IPv6 Address for Every Session and its Update Method

Kazuyuki NISHIDA†, Shingo ATA††, Hiroshi KITAMURA†††, and Masayuki MURATA†

† Osaka University 1-5 Yamadaoka, Suita-shi, Osaka 565-0871, Japan
†† Osaka City University 3-3-138 Sugimoto, Sumiyoshi-ku, Osaka-shi, Osaka 558-8585, Japan
††† NEC Corporation/University of Electro-Communications
2-11-5 Shibaura, Minato-ku, Tokyo 108-8557, Japan
E-mail: †{k-nisida,murata}@ist.osaka-u.ac.jp, ††ata@info.eng.osaka-cu.ac.jp, †††kitamura@da.jp.nec.com

Abstract　In this paper, we propose a new communication scheme, which utilizes IP address as a *key* of session, to realize a secure remote communication from outside network. In this scheme, IP address is an *access key* for a session by itself, and it varies for every session to reduce a security risk to be reused by anonymous parties. To realize this access scheme, we discuss on the listen model at the server, key updating, and key synchronizing issues. We also describe an implementation model of proposed scheme to show the feasibility.

Key words　Key address, Listening model, IPv6, Security, Unified Multiplex, Remote access

1. はじめに

2011 年現在、インターネットが直面する緊急かつ大きな問題として IPv4 (Internet Protocol version 4) アドレス枯渇問題がある [1], [2]。もはや IPv4 の新規アドレス割当の中止は時間の問題であり、早急かつ迅速な解決法が望まれる。IP アドレス枯渇の抜本的な解決は、広大なアドレス空間を持つ IPv6 アドレスへの移行以外にはなく、我が国においても通信事業者ならびにエンドユーザが速やかに IPv6 へ移行するためのアクションプランが策定され、着実に進められているところである。

しかしながら、IPv6 への移行はいわゆるネットワーク層のプロトコルの全面的な置き換えを意味する。単にアドレスの拡張という点だけをもって IPv6 へ移行するのではなく、IPv4 時代に生じていた問題を抜本的に解決し、IP 通信のさらなる進化を実現させる絶好の機会でもある [3]。

以上の背景のもと、我々はこれまで IPv6 ネットワークにおいてより直感的で、かつ通信の安心・安全性を向上させる通信アーキテクチャについて検討を行ってきた。Unified Multiplex 通信アーキテクチャ [4]〜[6] は、従来当たり前のように考えられていた「1 ノード–1 固定アドレス」という概念を根本的に見直し、「1 ノード–複数変動アドレス」という考え方を導入することで、単純な仕組みを提供しつつセキュリティやプライバシ

の向上を実現できる通信アーキテクチャである。

本稿では、Unified Multiplex の応用として、エンドユーザがセキュリティを意識することなく容易に外部からの所有機器のリモートアクセスを実現する方式を提案する。本稿で対象とするリモートアクセスとは、以下の要求条件を満たすものである。

- ユーザが所有あるいは制御できる機器を外部から容易にリモートアクセスできるようにする
- 不特定な第三者からその機器にアクセスされるというセキュリティリスクを可能な限り低減したい
- ファイアウォールのフィルタリング等、セキュリティの知識を要する設定をなるべく軽減したい
- サーバのアプリケーションを変更することなく、サーバのカーネル入れ替えおよびデーモンプロセスの起動だけで実現したい

近年、家電機器などさまざまな装置がネットワークに接続されつつある。それに伴い、所有するユーザがそれらの機器を外部から制御、管理したいというニーズが高まってきている。また、小規模のグループで排他的に資源（ディスクスペースなど）を共有したい場合、そのような資源は外部からアクセス可能な領域に設置することが必要となる。

上記の要求を満たすためには、当該機器を外部からアクセス可能な public な領域に設置し、かつ外部からのアクセスを待ち受けるという、TCP/IP 通信における「サーバ」の機能を有する必要がある。エンドユーザ側でサーバを設置する場合、そのようなファイアウォールに対してサーバごとに個別にフィルタリングルールを設定する必要があるなど、セキュリティに対する高い知識が要求される。さらに、適切なフィルタリングルールを記述したとしても、外部から不明な第三者がアクセスするリスクは常に存在するため、セキュリティに十分配慮しつつ慎重な運用が必要となる。

このように、現在のインターネットにおいて一般的なエンドユーザがサーバを外部ネットワークに設置することは、高いセキュリティの知識および運用能力が必要であり、容易ではない。したがって、大部分のエンドユーザが受け入れ可能な、簡便かつ安心な方法でのアクセス手法が大いに期待される。

IP ネットワークにおいて、外部アタックなどのセキュリティおよびプライバシを考慮した通信モデルについてはこれまでにもいくつか検討されている。[7] はセキュリティ向上のため、従来のポート番号決定方法を見直す手法が検討されている。また、[8], [9] ではプライバシ保護を考慮したアドレス生成手法、[10] ではより安全性の高いノード探索手法について検討されている。しかしながら、いずれの通信モデルも「1 ノード–1 固定アドレス」の仕組みにもとづくものである。我々は、IP アドレスに関するセキュリティ・プライバシ問題を抜本的に解決するためには「1 ノード–複数変動アドレス」が最も有効であると考えている。

本稿では、上記問題を解決するため、IPv6 のもつ広大なアドレス空間を有効に活用し、自身にしか分からないアドレスを使って通信することで、容易かつ安全にリモートアクセスできる手法を提案する。提案手法では、クライアントがサーバにアクセスするための IPv6 アドレスをエンドユーザが規定したルールに基づき生成する。接続のためのアドレスを第三者から隠匿することで、IPv6 アドレス自体をクライアント・サーバ間の秘密の共有鍵として扱うことができる。本稿では、リモートアクセスのための秘密の IPv6 アドレスを「鍵アドレス」と呼ぶ。

広大な IPv6 アドレス空間において、サーバの IPv6 アドレスが未知の場合、ランダムあるいは総当たりによってサーバにアクセスすることはほぼ不可能であるため、「サーバの鍵アドレスを教えない」だけで容易に不特定第三者からのアクセスを防ぐことが可能であり、別途フィルタやファイアウォールなど特別な仕掛けを必要としない。また、鍵アドレスを教えないというのは、携帯電話において「電話番号を変えて教えないことで迷惑電話から守る」と同様の考え方であり、エンドユーザにも広く受け入れられ易い。したがって、多くのユーザが容易に行うことが可能である。また、アドレスを知っているものだけがアクセスできるという単純なしくみであるため、エンドノード以外の第三の機器やサービスを必要としない、などの利点がある。

提案方式ではさらに安全性を向上させるため、通信セッションごとに独立した鍵（アドレス）を使用することを考える。これにより、一度通信に使用された、すなわち外部に知られたアドレスを第三者が用いてもアクセスすることができず、高い安全性を提供することができる。

このように提案方式は、アドレスをセッションごとに変化する秘密の共有鍵として用いることで、シンプルでありつつも高いセキュリティを提供できる。また、アドレスを秘密鍵として外部に教えないことは、セキュリティの知識を有しない一般ユーザにとっても直感的で理解しやすい防御策であり、リモートアクセスに対する安心感を高めることが可能である。しかしながら、このような方式を実現するためには、従来の IP の待ち受けモデル、アドレスの更新およびエンド間の鍵の同期などの技術課題が存在する。本稿では、これらの技術課題について詳細に分析し、それに実装モデルを示す。

以下 2. において、「鍵アドレス」によるリモートアクセスを実現するための、サーバの待ち受けモデルとアドレスの関係について考察する。次に、3. において提案するアクセス方式の概要について述べ、4. で必要となる機能の詳細について記述する。また、5. で実装モデルについて述べる。最後に 6. でまとめと今後の課題について述べる。

2. サーバの待ち受けモデルと鍵アドレスの関係

アドレスをセッションごとに変化させる鍵として用いるためには、ソケットの状態とアドレスの状態の関係を再検討する必要がある。本章では、提案方式における待ち受けモデルと鍵アドレスの関係について考察し、既存の IP ネットワーク（以降ではレガシー環境と呼ぶ）との相違点について議論する。

2.1 レガシー環境における待ち受けモデル

まず、レガシー環境におけるサーバの待ち受けモデルを図 1 に示す。レガシー環境では、IP アドレスはインターフェース起

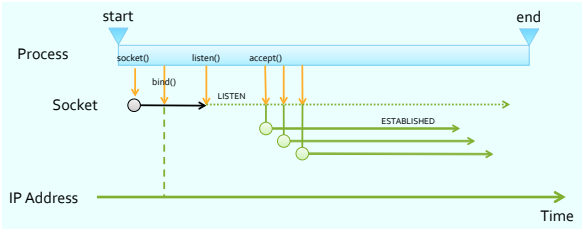

図 1　既存の IP ネットワークにおける待ち受けモデル

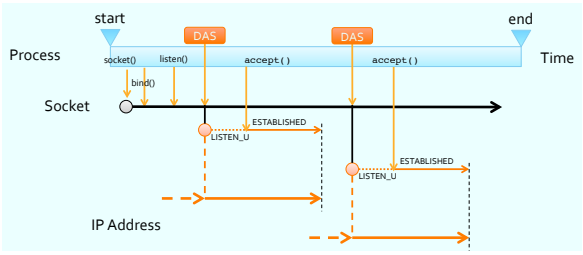

図 2　提案方式における待ち受けモデル

動時に割り当てられ、ネットワークの接続状態が変化しない限りほぼ永続的に使用される。通信プロセスは内部で `socket()` システムコールによりソケットを作成し、`bind()` によってアドレスに関連づけられる。ここで関連づけられるアドレスは特定のインターフェースの IP アドレスか、ノード上のすべてのインターフェースで待ち受けることを表す `INADDR_ANY` のいずれかである。次に `listen()` によって待ち受けを開始する。このとき、レガシー環境のサーバは複数からの接続を同時に待ち受ける「多重同時接続待ち受け」が採用されている。クライアントからの `connect()` によって `accept()` が呼び出されると、クライアントとの通信用に新たなソケットが生成される。このとき新しいソケットに関連づけられるアドレスは実際に接続された IP アドレスである。クライアントからの複数の `connect()` に対しては、それぞれごとに新しいソケットが生成され、同じ IP アドレスが関連付けされる。

2.2　提案方式における待ち受けモデル

一方、鍵アドレスを使用する提案方式では、鍵を表す IP アドレスは通信セッションごとに異なり、通信が終了した IP アドレスは速やかに消去される必要がある。これを実現するため、サーバの待ち受けモデルを従来の「多重同時待ち受け」から「アドレスごとの単一待ち受け」に変更する必要がある。これを実現するため、提案方式では Unified Multiplex の DAS (Delayed Address Setting)、および LISTEN_U 状態を用いる。

図 2 に、提案方式における待ち受けモデルを示す。レガシー環境と同様 `listen()` により待ち受けを開始するが、単に `listen()` を行っただけではクライアントからの接続は受け付けられない。この時点では待ち受けのためのアドレスがまだ決められていないためである。待ち受け時点で通信セッションごとに異なるアドレスを割り当てるため、Unified Multiplex では以下の処理を行う。

- セッション専用のアドレスを事前準備する
- DAS (Delayed Address Setting) により、ソケットにアドレスを関連づける。具体的には、クライアントとの通信用のソケットを 1 つ作成し、DAS で指定したアドレスを関連付けする。このとき、`accept()` できるクライアント数は 1 つのみである。複数のクライアントに対する待ち受けを行いたい場合は、必要な数だけ DAS を実行する。
- `accept()` によりクライアントからの `connect()` を受け付ける。このときは新しいソケットは生成せず、ソケットの状態を ESTABLISHED に変更するのみとなる。
- ソケット通信が終了した段階で、関連づけられた IP アドレスを消去し、以降の使用を無効とする。

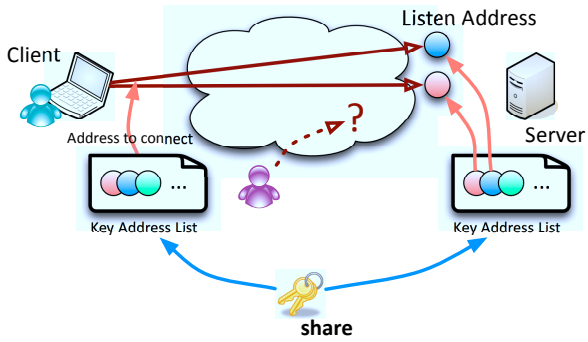

図 3　提案手法概要

3.　アーキテクチャ概要

本章では、提案するリモートアクセス手法に関する概要について述べ、必要な機能について説明する。各機能の詳細については次章で述べる。

図 3 に提案するアクセス手法の概要を示す。アクセスのための手順は以下の通りである。

（1）　あらかじめ、サーバおよびクライアント間で接続のために必要となる鍵アドレスのリスト (Key Address List; KAL) を共有する。鍵アドレスはプレフィックス部を固定し、インターフェース識別し部分を変化させて生成する。

（2）　サーバは通信セッションごとに異なる待ち受けアドレスを鍵アドレスリストから選択したアドレスに設定する。

（3）　クライアントは、サーバに接続したい時に、鍵アドレスリストから現在サーバが待ち受け中のアドレスを選択し、そのアドレスに対して接続を行う。

（4）　サーバは通信セッションが完了したら使用したアドレスを破棄する。

（5）　新しい通信セッションを行う場合は、サーバは鍵アドレスリストから新たなアドレスを待ち受けアドレスとして使用する。

クライアントがサーバに接続するためには、現在サーバが待ち受けを行っているアドレスを接続前に知っていなければならない。IPv6 のインターフェース識別子は通常 64 ビットであることから、クライアントがサーバの待ち受けアドレスを知らない場合、誕生日問題を考慮しても試行錯誤的にそのアドレスを知ることは事実上不可能である。このため、鍵アドレスリストを共有していない第三者はサーバに接続することができない。また、第三者がクライアント・サーバ通信をモニタリングして

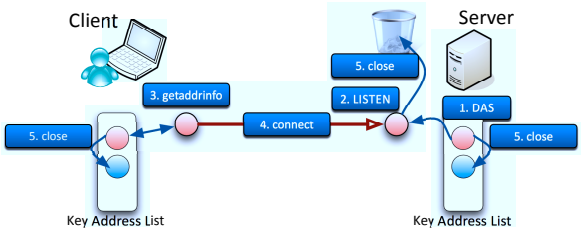

図 4　鍵アドレスを用いた通信の動作シーケンス

通信セッションで使用されたアドレスを入手できたとしても、入手できたアドレスは現在通信中のセッション専用に割り当てられており（すなわち、新たな接続は受け付けられない）、またセッション終了後は直ちにアドレスが破棄されることから、入手したアドレスを使用してサーバと通信を行うこともできない。以上の方法によりサーバ側でファイアウォールやフィルタリングを行うことなく、クライアントだけが接続可能な環境が構築できる。

ここで特筆すべき点は、上記アクセス手法において、サーバ・クライアント間では鍵アドレスリストの共有以外は一切のネゴシエーションを行わないことである。特にクライアントが遠隔にいる場合でも、クライアントは現在サーバが使用している待ち受けアドレスを知ることが可能である。

4.　変動型待ち受けアドレスを実現するためのアドレス生成、更新、同期処理

図 4 にサーバおよびクライアントにおける接続プロセスと使用する鍵アドレスに関する動作シーケンスを示す。

（1）　サーバは鍵アドレスリストから待ち受けアドレスに使用するアドレスを 1 つ取得し、待ち受けソケットに関連づける。関連づけは Unified Multiplex の DAS (Delayed Address Setting) により行う。

（2）　DAS 完了後、サーバの待ち受けソケットは `LISTEN_0` から `LISTEN_U` 状態に変更され、クライアントからの接続を受け付ける。

（3）　クライアントは `getaddrinfo()` によってサーバのホスト名に対する IP アドレスを取得する。このとき、`getaddrinfo()` で返されるアドレスを、サーバが待ち受けに使用している鍵アドレスにする。

（4）　クライアントは、`getaddrinfo()` で得たアドレスを使用して、サーバに接続を行う。接続後、通常通り通信を行う。

（5）　通信セッションが完了した段階 (`close()`) で、サーバ・クライアントはそれぞれ以下の処理を行う。

（a）　サーバは、使用した鍵アドレスをインターフェースから削除し、invalid 状態にする。

（b）　サーバ、クライアント双方で使用する鍵アドレスの参照ポインタを一つ移動させる。

以下では、各手順において重要となる待ち受けアドレスの生成、更新、および同期処理について詳細に述べる。

4.1　サーバにおける待ち受けアドレス更新処理

待ち受けアドレスの更新タイミングは、サーバがどのように

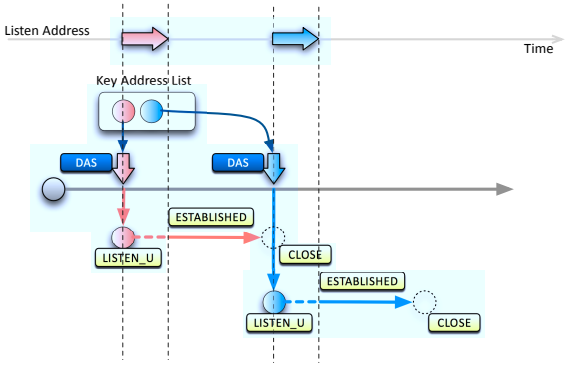

図 5　待ち受けアドレスとソケット状態遷移の関係（シリアル更新型）

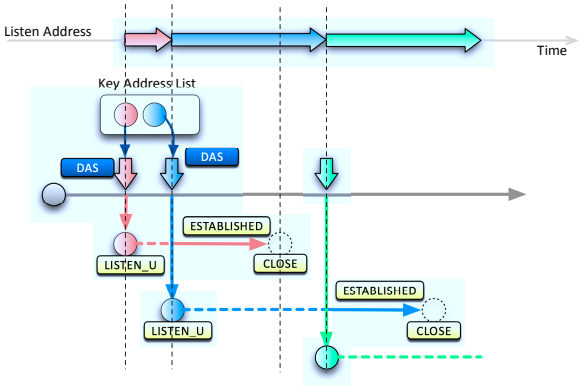

図 6　待ち受けアドレスとソケット状態遷移の関係（パラレル更新型）

待ち受けを行うかによって異なる。その方法は大きく「シリアル更新型」「パラレル更新型」の 2 種類に分類することができる。シリアル型は、サーバがクライアントと同時に通信できるセッション数を 1 に制限したものである。シリアル更新型における待ち受けアドレスとソケット状態遷移の関係を図 5 に示す。シリアル更新型では、まず 1 回の DAS により、鍵アドレスで待ち受けソケットを 1 つ作成する。そして通信セッションが終了した段階で、ソケットが削除されることから、代わりの待ち受けソケットを DAS により作成する。したがって、通信可能なソケットは常に 1 となり、クライアントとの通信セッションが確立中は他の待ち受けは行わない。

一方、パラレル更新型はソケットが `ESTABLISHED` になった段階で、新たな待ち受けソケットを作成するモデルである。アドレスと状態遷移の関係を図 6 に示す。シリアル更新型とは異なり、クライアントからの `connect()` により待ち受けソケットが `LISTEN_U` から `ESTABLISHED` に遷移した段階で新たな DAS を実行し、別の待ち受けソケットを作成する。このため、サーバには常に待ち受けソケットが 1 つ存在することになる。

なおシリアル更新型、パラレル更新型とも、上記シーケンスを多重化することにより複数の待ち受けが可能である。N 個の多重化の場合、シリアル更新型は待ち受けと通信セッション確立中のソケット数の合計が N となり、パラレル更新型は同時に待ち受け可能ソケット数が N となる。

4.2　アドレス同期と再同期処理

パケットロスや機器の故障が発生しないなど、理想的な通信環境下では、サーバ・クライアントともに通信セッションの開

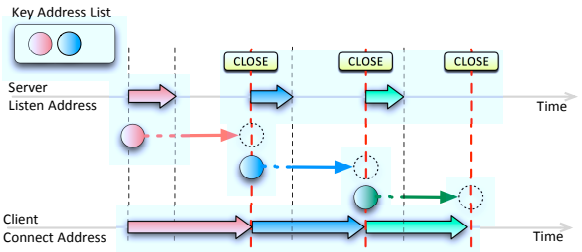

図 7　理想環境下におけるアドレスの同期

始と終了が同期していることから、鍵アドレスリストの参照ポインタを通信セッションごとに 1 つずつずらしていくことで次の通信セッションのための待ち受けおよび接続アドレスを知ることが可能である。図 7 は、サーバ・クライアントにおいて待ち受けおよび接続に使用するアドレスの時間的変化を示したものである。この図のように、サーバおよびクライアントが離れている環境下においても、通信の開始と終了時点でタイミングを同期させることが可能であるため、セッション終了時を同期タイミングとしてアドレスの変更を行うとアドレスの同期も行うことができる。

しかしながら、実環境下では様々な理由により、アドレスの同期が必ずしも行われないことがある。以下に例を示す。

- サーバあるいはクライアントにおいて機器の再起動、サスペンド、アプリケーションの停止などが発生することで、鍵アドレスリストあるいはリスト中の現在の参照位置に関する情報が消失した場合。
- 経路中に通信障害が発生し、クライアントから接続ができなかった場合。このときは接続に使用されたアドレスを再度用いることはセキュリティ上好ましくないため、接続に失敗した場合は使用するアドレスを更新する方が望ましい。また、アプリケーション側からの再接続の場合、通常と同様にアドレスが更新されることになる。しかしながら通信に失敗したという事実をサーバ側が知る方法はないため、サーバ側のアドレスは更新されない。
- 非常にまれな確率で第三者からの接続があった場合。
- サーバに通信したいクライアントノードが同時に複数存在する場合。クライアント間の通信状況をお互い把握することができないため、通信セッションによる同期を行うことができない。

通信セッションの開始・終了による同期が正しく動作しない場合、そのままでは鍵アドレスリストの参照位置にずれが生じたままとなり、以降の接続がすべて失敗する可能性がある。これを解決するため、本稿では通信セッションの開始終了以外に時刻情報に基づく同期を組み合わせて用いる。

図 8 に、接続失敗による同期のずれと時刻情報にもとづく再同期の流れを示す。クライアント側でネットワーク障害による接続の失敗が発生すると、クライアントはアドレスリスト内の次の鍵アドレスを用いて再接続を行おうとする。このため、同期のずれが発生する。そこで、セッションの同期とは独立して、ある時間間隔にもとづいて再同期を行う。クライアントでは、

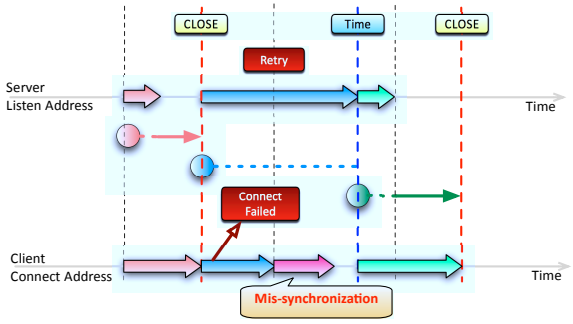

図 8　接続失敗による同期のずれと時刻情報にもとづく再同期

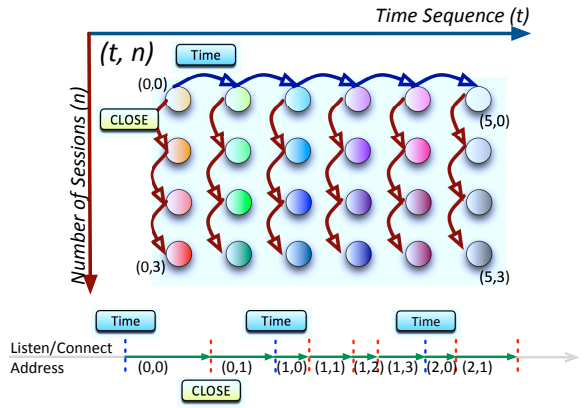

図 9　アドレス生成系列

時刻情報による再同期までは同期ずれが継続し、サーバに接続することはできないが、時刻による再同期後はアドレス情報が同期され、再度接続が行えるようになる。

4.3　待ち受けアドレス生成処理

前節のアドレスを考慮した場合、待ち受けアドレスの同期は

- セッションの終了時（セッション毎同期）
- 一定時間後（時刻再同期）

の 2 段階で行われる。同期の方法によって鍵アドレスとして使用できる情報が異なるため、アドレスの生成系列は図 9 で示すとおりとなる。

すなわち、生成されるアドレス系列は一次元的にはならず、時刻同期による系列（横軸）と、セッション毎同期による系列（縦軸）の二次元系列となる。本稿では、同期時刻 t における n 番目の通信セッションのための鍵アドレスを (t, n) で表す。具体的には、最初の通信セッションは $(0, 0)$ のアドレスを使用して行われ、以降 $(0, 1), (0, 2), (0, 3) \ldots$ と変化する。

セッション毎系列は時刻再同期のタイミングでリセットされる。すなわち $(0, i)$ である状態は、時刻による再同期が発生した段階で $(1, 0)$ にリセットされ、以降 $(1, 1), (1, 2), (1, 3) \ldots$ と変化する。

図 10 にアドレス系列を生成するアドレスジェネレータの実装例を示す。もちろん、アドレス系列生成アルゴリズムを秘密の共有情報とする方法も考えられるが、生成アルゴリズムを共通化し、生成の鍵となるパスフレーズを秘密の共有情報とする方が運用上は簡便である。また、生成されるアドレスは外部からの類推が困難となるよう、ランダム性が高い（すなわち、ア

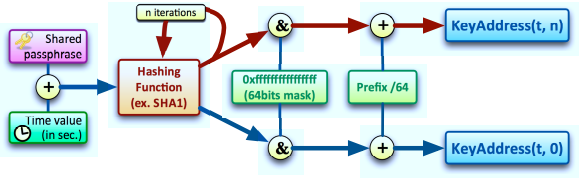

図 10　アドレスジェネレータ

ドレス空間に一様に分布する）ことが望ましい。したがって、キーの生成にはハッシュ関数 (SHA1 など) を用いることが有用である。

以下の例では、サーバ・クライアントでパスフレーズを秘密情報として共有し、時刻による $(t, 0)$ のアドレス系列の生成は、パスフレーズと時刻（t を秒で変換したもの）を合成した情報をハッシュの入力として用いる。セッション毎同期のアドレス系列は (t, n) は、 $n-1$ 番目のセッションで使用されるアドレスをハッシュの入力として繰り返し計算を行うことで生成する。そして、得られたハッシュ値の下位 64 ビットを計算し、さらにプレフィックス部とビット和をとることで、最終的な鍵アドレス (t, n) を得る。

5. 実装モデル

図 11 に、提案するアクセス手法の実装モデルを示す。サーバとクライアント間ではあらかじめパスフレーズを秘密の共有情報として保有するものとする。

アドレスジェネレータはパスフレーズと時刻情報、および同期時刻以降の通信セッション数をもとに、鍵アドレスの系列を生成する。生成はサーバ・クライアントで独立して行われる。

アドレスジェネレータから生成されたアドレスをもとに、サーバ側では DAS によるアプリケーションの通信ソケットへのアドレス割り当てと待ち受け、クライアント側ではサーバの FQDN に対する接続アドレスの更新を行う。FQDN に対応した鍵アドレスを適宜更新する必要があるため、DNS masquerade の機能を用いる。

次に、クライアントアプリケーションからの `getaddrinfo()` によって、サーバの接続アドレスを取得し、`connect()` により接続を行う。

ソケットの状態は Routing Socket (PF_ROUTE) を用いて監視され、図 5 および 6 の状態遷移にもとづき、セッション毎同期をアドレスジェネレータに通知する。一方、サーバおよびクライアント間は NTP 等により時刻同期を行い、あらかじめ指定した同期時刻にもとづいてアドレスジェネレータに時刻再同期を通知する。通知を受け取ったアドレスジェネレータは新しいアドレス系列に遷移し、次の通信セッションのための鍵アドレスを生成する。

6. まとめと今後の課題

本稿では、アドレスを鍵として用い、通信セッションごとに変動させることで、外部ネットワークを介したリモートアクセスをセキュリティを意識することなく実現するアクセス手法を

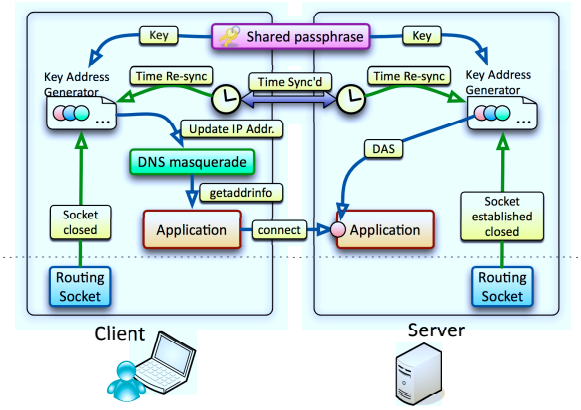

図 11　実装モデル

提案した。

今後は、広域網における動作検証を通じた安全性の確認、および時刻同期間隔等のパラメータに関する検討を行っていく予定である。

文　献

[1] ICANN, “Available pool of unallocated IPv4 internet addresses now completely emptied,” *Press Release, available at* http://www.icann.org/en/news/releases/release-03feb11-en.pdf, February 2011.

[2] APNIC, “IPv4 exaustion,” *available at* http://www.apnic.net/community/ipv4-exhaustion/, 2011.

[3] D. Thaler, “Evolution of the IP model,” *RFC 6250*, May 2011.

[4] 北村 浩, 阿多 信吾, 村田 正幸, “セッション毎に専用の IP アドレスを用いることが可能にする通信アーキテクチャの進化 - Unified Multiplex -,” **電子情報通信学会技術研究報告**, vol. 109, no. 449, pp. 405–410, 2010.

[5] S. Ata, H. Kitamura, and M. Murata, “Architectural design of unified multiplex communications for one-time use of IP addresses,” in *Proceedings of 4th IFIP International Conference on New Technologies, Mobility and Security (NTMS 2011)*, (Paris, France), pp. 1–4, February 2011.

[6] K. Sakakima, S. Ata, and H. Kitamura, “Anonymous but traceable IP address-based communication system.,” in *Proceedings of NETCOM 2009*, pp. 259–264, December 2009.

[7] M. V. Larsen and F. Gont, “Transport protocol port randomization recommendations,” *Internet-Draft, draft-larsen-tsvwg-port-randomization-09*, August 2010.

[8] T. Narten and R. Draves, “Privacy extensions for stateless address autoconfiguration in IPv6,” *RFC 4941*, September 2007.

[9] T. Aura, “Cryptographically generated addresses (CGA),” *RFC 3972*, 2005.

[10] J. Arkko, J. Kempf, B. Zill, and P. Nikander, “Secure neighbor discovery (send),” *RFC 3971*, 2005.